Preserving Caller Anonymity in Voice-over-IP Networks

Mudhakar Srivatsa[†], Ling Liu[‡] and Arun Iyengar[†]
IBM T.J. Watson Research Center, Yorktown Heights, NY - 10598[†]
College of Computing, Georgia Institute of Technology, Atlanta, GA - 30332[‡]
{msrivats, aruni}@us.ibm.com, lingliu@cc.gatech.edu

***Abstract*— Applications such as VoIP need to provide anonymity to clients while maintaining low latency to satisfy quality of service (QoS) requirements. Existing solutions for providing anonymity such as mix networks are not well suited to applications like VoIP, SSH, and gaming which require low communication latency. This paper investigates the problem of on-demand construction of QoS sensitive routes on anonymizing networks using the VoIP application. We first describe triangulation based timing analysis attacks on shortest path route set up protocols. We show that even when a small fraction ($\sim$1%) of the network is malicious, the adversary can infer the source (caller) with reasonably high probability. Second, we describe random walk based route set up protocols that significantly improve anonymity while satisfying latency-based QoS guarantees. We describe a prototype implementation of our proposal and show that our protocols can significantly reduce the probability of inferring the caller. We present a detailed experimental evaluation to demonstrate our attacks and quantify the performance and scalability of our guards.**

I. Introduction

Many applications such as VoIP need to provide anonymity to clients using the application. Existing approaches typically use mix networks [10], [17], [12], [16], [8] which provide good anonymity for high latency communications by routing network traffic through a number of nodes with random delays and random routes. However, emerging applications such as VoIP[1], SSH, online gaming, etc have additional quality of service (QoS) requirements that are hard to be accommodated by such mix networks; for instance ITU (International Telecommunication Union) recommends up to 250ms one-way latency for interactive voice communication[2].

[1]VoIP's share of worldwide voice traffic has grown from 12.8% in 2003 to an estimated 75% in 2007 [6]

[2]A case study [29] indicates that latencies up to 250ms are unperceivable to human users, while latencies over 400ms significantly deteriorate the quality of voice conversations.

Several authors have pointed out that low latency applications on mix networks may be vulnerable to timing analysis attacks [30], [35], [28]. In this paper, we investigate trade offs between QoS guarantees and anonymity using VoIP as a sample application. A VoIP network typically consists of a core proxy network and a set of clients that connect to the edge of this proxy network (see Figure 1). We show how the identity of VoIP callers can be identified via timing attacks. We then present solutions for preserving anonymity while satisfying QoS requirements.

In particular, this paper investigates the problem of on-demand construction of QoS sensitive routes on anonymizing networks and makes two contributions. First, we describe triangulation based timing analysis attacks on a peer-to-peer broadcast based shortest route set up protocol. Unlike previous timing analysis attacks [33], [36], [38], [32], [9], [13], [31] that use inter-packet timing characteristics, our timing analysis attacks focus exclusively on the execution times of different stages in the route set up protocol. We show that while the VoIP route set up protocol meets the QoS requirement (by setting up the shortest path), it is vulnerable to timing analysis that can reveal the identity of the caller with non-trivial probability even when only a small fraction of the network nodes are malicious ($\sim$1%).

Second, we develop and implement three solutions to improve the resilience of route set up protocols to provide anonymity while satisfying latency-based QoS guarantees. First, we show that naively adding random delays to network latencies does not alleviate the problem. We then show that a pure random walk based route set up protocol can significantly reduce the probability of inferring the caller, although it may blatantly violate QoS requirements by setting up routes with unbounded latencies. We describe two hybrid route set up protocols that combine random walk and shortest route set up protocols with the goal of providing resilience to

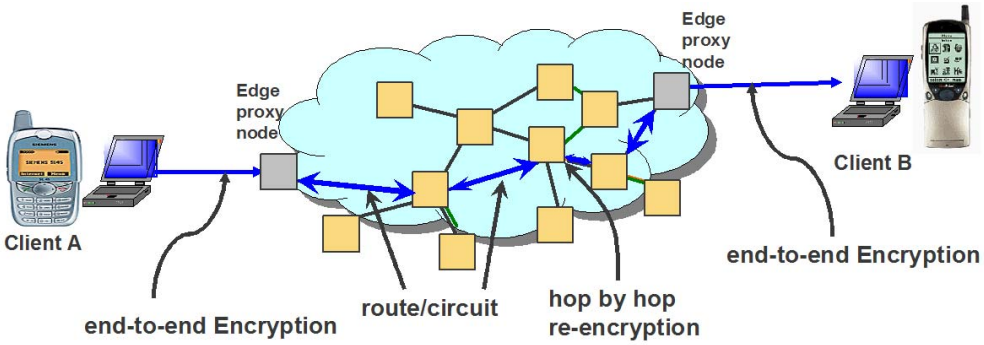

Fig. 1. Anonymizing VoIP Network

triangulation based timing attacks, while satisfying QoS requirements.

The rest of this paper is organized as follows. We describe a VoIP network model in Section II. We describe the triangulation based timing attacks in Section III followed by our guards in Section IV. We describe a prototype implementation and present detailed experimental evaluation to demonstrate our attacks and quantify the performance and scalability of our guards in Section V. We discuss related work in Section VI and conclude the paper in Section VII.

II. Preliminaries: VoIP Route Set Up Protocol

VoIP applications typically use two main protocols: a Route Set Up protocol for call setup and termination, and a Real-time Transport Protocol (RTP) for media delivery. Our work is relevant to peer-to-peer VoIP protocols (such as Skype [5]) which use route set up protocols similar to the one described in this section. The route set up protocol allows a caller to search for a receiver (identified by a URL, e.g.: sip:me.xyz.com) and sets up the *shortest* route (also called circuit) to the receiver node. RTP may be used to carry voice traffic between the caller and the receiver along an established bi-directional voice circuit.

The VoIP route set up protocol typically operates in four steps. First, the `initSearch` initiates a route set up request from a VoIP client src. Second, the `processSearch` processes a route set up request at some node on the VoIP network. Third, the `processResult` processes the results of a route set up request at some node on the VoIP network. Fourth, the `finSearch` concludes the route set up procedure. We now describe these four operations in detail, which are important for understanding triangulation based timing attacks discussed in the next section.

`initSearch`. A VoIP client src initiates a route set up for a receiver dst by broadcasting $search$ ($searchId$, $sipurl = dst.sipurl$, $ts = curTime$) to all nodes $p \in ngh(src)$, where $ngh(src)$ denotes the neighbors of node src in the VoIP network. The search identifier $searchId$ is a long randomly chosen unique identifier, and ts denotes the time stamp at which the route set up request was initiated.

`processSearch`. Let us suppose p receives $search$ ($searchId$, $sipurl$, ts) from its neighbor q. If p has seen $searchId$ in the recent past then it drops the request. Otherwise, p checks if $sipurl$ is the URL of a VoIP client connected to p. If yes, p returns its IP address using $result$ ($searchId$, p) to q. Otherwise, p broadcasts $search$ ($searchId$, $sipurl$, ts) to all $p' \in ngh(p) - \{q\}$ and caches $\langle searchId, sipurl, q\rangle$ in its recently seen list. Note that p' has no knowledge of where the route set up request is initiated.

`processResult`. Let us suppose p receives $result(searchId, q)$ from q. Note that p has no knowledge as to where the search result was initiated. p looks up its cache of recently seen search queries to locate $\langle searchId, sipurl, prev\rangle$. p adds a routing entry $\langle sipurl, q\rangle$ and forwards $result$ ($searchId$, p) to $prev$.

`finSearch`. When src receives $result$ ($searchId$, q) from q, it adds a routing entry $\langle dst, q\rangle$ to its routing table. After a successful route set up, the clients src and dst exchange an end-to-end media encryption key and switches to the media delivery phase.

The route set up protocol includes the following security features: (i) All communication between a node and its neighbor is encrypted with a shared symmetric key. (ii) While the above protocol does not directly reveal src, it exposes dst (namely, $dst.sipurl$). This is usually fixed by replacing plain-text $dst.sipurl$ by an encryption of ($searchId$, ts, $dst.sipurl$) using dst's public key[3]. A node p checks if the request matches $sipurl$

[3]Public Keys are obtained out-of-band, say, using a trusted certification authority

by attempting to decrypt $dst.sipurl$ using its private key[4]. Additionally, the caller may add its ID ($src.sipurl$) signed using src's private key (and encrypted using dst's public key); this allows a receiver dst to respond with a search result only after verifying the caller ID. (iii) When the destination node dst receives the route set up request (`processSearch`), it adds a random delay (0-250ms) before sending the search result to its neighbor $prev$; this increases the route set up latency but not the route latency itself and makes it hard for $prev$ to infer that dst is indeed the destination using the time elapsed between forwarding the route set up request and receiving the search result. (iv) Finally, note that if all neighbors of a node n are malicious, then they can infer if n is indeed the source or the destination of a VoIP call. However, we investigate timing analysis attacks with significantly fewer malicious nodes ($\sim$1% of the network size) that can identify the caller of *any* VoIP call.

III. Caller Identification Attacks

In this section, we present triangulation based timing attacks on the VoIP route set up protocol. Similar to most other threat models for anonymizing networks, we assume that some network nodes (VoIP proxies) may be malicious and may collude with one another. The attacks require at least some knowledge of the VoIP network topology. We present techniques to exploit network maintenance protocols to discover network topology in Section III-D. Triangulation based timing attacks operate in three steps: `candidate caller detection`, `candidate caller ranking`, and `triangulation`. In the `candidate caller detection` step, malicious nodes passively observe the time instants at which they receive the route set up requests. Each malicious node (independently) uses this information and the topology of the VoIP network to deduce a set of candidate callers. In the `candidate caller ranking` step, malicious nodes associate a score with each such candidate caller s that denotes the likelihood of s being the actual caller. In the `triangulation step`, two or more malicious nodes combine their sets of candidate callers to obtain a much more *concise* and yet *precise* list of candidate callers.

We describe three timing attacks on the route set up protocol with increasing sophistication: deterministic triangulation, statistical triangulation and differential triangulation. First, we illustrate triangulation based timing attacks in a simplified setting to highlight the key properties of such attacks. Concretely, we make two assumptions: (i) the network link latencies are deterministic, and (ii) all nodes in the network have a tightly synchronized clock. Second, we relax the first assumption by showing how statistical triangulation attacks can operate on statistical link latencies (for arbitrary probability distributions) using the notion of statistical shortest paths. Third, we develop differential analysis that can tolerate arbitrary clock skews. While the first two attacks can be thwarted by simply removing the timestamp field `ts` from the route set up request, the third attack is completely agnostic to `ts`.

A. Deterministic Triangulation

In this section, we describe the properties of the broadcast based route set up protocol. Then, we use the shortest path property to develop triangulation based timing analysis attacks assuming deterministic link latency and tight clock synchronization.

1) Route Set Up Protocol Properties: By assuming deterministic latencies of the VoIP network, we observe two important properties of the route set up protocol:

Lemma 3.1: Let us suppose that the VoIP network link latencies are deterministic and that all the network nodes have tightly synchronized clocks. Let $dist(x, y)$ denote the length of the shortest path between nodes x and y. The route set up protocol satisfies the following properties:

`(i)` The protocol establishes the shortest route between the two nodes src and dst.

`(ii)` Any node p on the network that receives the route set up request originated from node src can estimate $dist(src, p)$.

Proof: A sketch of the proof is provided here. When p first receives a route set up request, that request must have traversed the shortest route between src and p. In step `processSearch`, a peer p can estimate $dist(src, p)$ using the time stamp ts on the route set up request and time instant at which the request was received by peer p. Also, if the peer p received the first route set up request from its neighbor q, then the shortest route from src to p is via q. Using mathematical induction on the number of hops traversed by a route set up request, one can show that the route set up step (`processResult`) builds the shortest route src to dst. ■

Figure 3 illustrates the protocol with $src = p_1$ and $dst = p_7$. The links in the VoIP network are labeled with link latencies (assumed to be deterministic). We label each

[4] dst avoids replay attacks using $searchId$ and ts

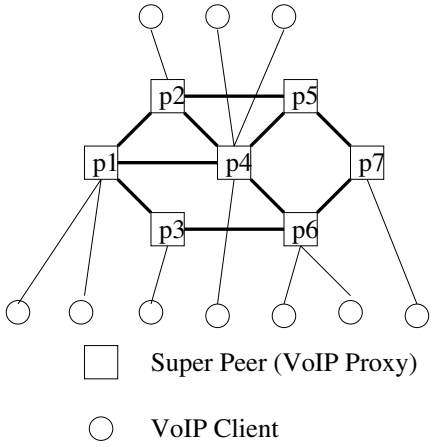

Fig. 2. VoIP Network

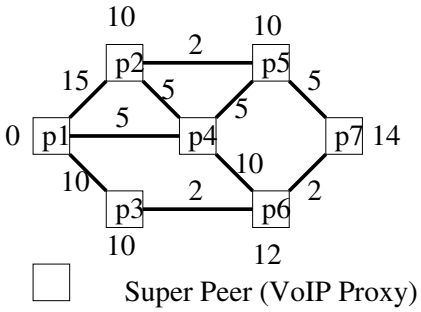

Fig. 3. Shortest Route Set Up Protocol

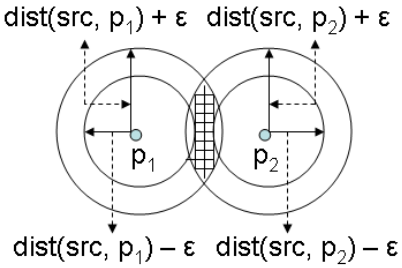

Fig. 4. Triangulation Attack Illustration: Caller Lies in Shaded Region

node with the time instant at which it received the first route set up request starting with peer p_1 at time $t = 0$. Evidently, the protocol establishes the shortest route $p_1 \leftrightarrow p_3 \leftrightarrow p_6 \leftrightarrow p_7$.

2) Deterministic Triangulation Attack: The *deterministic shortest path triangulation* attack easily follows from the properties of the route set up protocol. The attack operates in three steps. The `candidate caller detection` step operates as follows. Let p be a malicious node that received a route set up request originating from src at time ts. The malicious node p can compute $dist(s, p)$ for all nodes s in the network using Dijkstra's shortest path algorithm. Given the time instant t_p at which the request was received by peer p, p can estimate $\widehat{dist}(src, p) = t_p - ts$. Node p compiles a list of potential callers, denoted by $S(p)$, such that for any $s \in S(p)$, $|dist(s, p) - \widehat{dist}(src, p)| < \epsilon$, where ϵ is a tuneable parameter. In the `candidate caller ranking` step, the peer p computes the score for every node $s \in S(p)$ as $score_p(s) = \frac{1}{|dist(s,p) - \widehat{dist}(src,p)|+1}$, and sorts all nodes in $S(p)$ in the descending order of their scores. In the `triangulation` step, a set of colluding malicious nodes $p_1, p_2, \cdots, p_n$, compute $score(s) = \frac{\sum_{i=1}^{n} score_{p_i}(s)}{n}$ for every s in $S = \cap_{i=1}^{n} S(p_i)$. Figure 4 illustrates a triangulation with two malicious nodes p_1 and p_2.

In addition to the shortest path, a node p may receive duplicate route set up requests along alternate paths. This information can be used in a *deterministic multi-path triangulation attack* as follows. Let $ngh(p)$ denote the set of neighbors of node p. Initialize set $I = ngh(p)$. For every node $v \in I$, we compute the shortest path from src to v. If node p belongs to the shortest path from src to v, then we remove node v from the set I. One can show that the number of copies of route set up requests received by node p is equal to the cardinality of the resultant set I. The time instant at which p receives a route set up request from a node $v \in I$ is given by $dist(src, v)$ + e_{vp}, where $dist(src, v)$ is the length of the shortest path from src to v and e_{vp} denotes the latency of the edge from v to p. One can sort these time instants in ascending order and use $dist^{\rho}(src, p)$ to denote the time instant at which the ρ^{th} copy of a route set up request from src reaches p ($1 \leq \rho \leq |I|$); we set $dist^{\rho}(src, p) = \infty$ for $\rho > |I|$. Now, we compute the score for a candidate caller s as $score_p(s) = \sum_{\rho} \frac{1}{|dist^{\rho}(s,p) - \widehat{dist}^{\rho}(src,p)|+1}$, where $\widehat{dist}^{\rho}(src, p) = t_p^{\rho} - ts$ and t_p^{ρ} denotes the time instant at which the ρ^{th} request is received by p.

B. Statistical Triangulation Attack

In this section, we use the notion of statistical shortest paths [11] to relax the deterministic link latency assumption. Similar to [11], we model link latencies as independent distributions characterized by mean μ_e and variance σ_e^2. Assuming that the link latencies are independently distributed, the length of a path $P = \{e_1, e_2, \cdots, e_r\}$, is given by ($\mu_P = \sum_{i=1}^{r} \mu_{e_i}$, $\sigma_P^2 = \sum_{i=1}^{r} \sigma_{e_i}^2$). The notion of a statistical shortest path is defined using two operators $\preceq$ and $\parallel$ on path lengths: $(\mu_1, \sigma_1^2) \preceq (\mu_2, \sigma_2^2)$ holds if $\mu_1 \leq \mu_2 \wedge \sigma_1^2 \leq \sigma_2^2$, and the $(\mu_1, \sigma_1^2) \parallel (\mu_2, \sigma_2^2)$ holds if $\mu_1 \leq \mu_2 \wedge \sigma_1^2 \geq \sigma_2^2$ or $\mu_2 \leq \mu_1 \wedge \sigma_2^2 \geq$. The goal of a statistical shortest path algorithm is to compute a list of *Pareto-optimal* shortest path lengths from a node src to a node p on the network. A set of path lengths $d_1 = (\mu_1, \sigma_1^2)$, $d_2 = (\mu_2, \sigma_2^2)$, $\cdots$, $d_m = (\mu_m, \sigma_m^2)$ is Pareto-optimal if for all other path lengths d, $d_i \preceq d$ for all i and $d_i \parallel d_j$ for all i and j such that $i \neq j$ and $1 \leq i, j \leq m$.

Now we sketch a statistical triangulation attack in terms

of the three step process. In the `candidate caller detection` step, a malicious node p computes a set of Pareto-optimal set of statistical shortest distances to all nodes v ($dist_p[v]$) in the network. Then, p marks v as a candidate caller if $|\widehat{dist}(src,p) - \mu_d| < \epsilon$, where $\widehat{dist}(src,p) = t_p - ts$ and there exists $d \in dist_p[v]$ such that $d = (\mu_d, \sigma_d^2)$. The second step is to compute a likelihood score for each candidate caller v. If the link latency distributions were Gaussian, then one can show that the path latencies are Gaussian as well. Given $d = (\mu_d, \sigma_d^2) \in dist_p[v]$, one can use the Gaussian distribution to determine the likelihood that d matches the observation $\widehat{dist}(src,p) = t_p - ts$ as $l_d = Gauss_{\mu_d,\sigma_d^2}(\widehat{dist}(src,p))$, where $Gauss$ denotes a Gaussian distribution. For other probability distribution functions, we compute an approximation to this likelihood using Chebyshev's inequality [21]: $Pr(|X-\mu| \geq k\sigma) \leq \frac{1}{k^2}$. Hence, given $\widehat{dist}(src,p)$ and $d = (\mu_d, \sigma_d^2) \in dist_p[v]$, we approximate the likelihood score as $l_d = \frac{1}{k^2}$, where $k = \frac{|\widehat{dist}(src,p)-\mu_d|}{\sigma_d}$. Finally, p assigns a score $score_p(v)$ for a candidate caller v as the maximum likelihood l_d over all $d \in dist_p[v]$. In the `triangulation` step, we simply compute an aggregate score[5] for a candidate caller v as $score(v) = \frac{\sum_{i=1}^{n} score_{p_i}(v)}{n}$.

C. Differential Triangulation Attack

Both the deterministic and statistical triangulation assumed clock synchronization between src and p when estimating $\widehat{dist}(src,p) = t_p - ts$. In general the clock skew is not likely to be unbounded. Hence, one can use a statistical model for clock skew (quantified by μ_{skew} and σ_{skew}^2) to enhance the statistical triangulation attack. However, an obvious way to thwart deterministic and statistical triangulation attacks is to eliminate the time stamp `ts` from the route set up request. The differential triangulation attack can tolerate arbitrary clock skew and is completely agnostic to the time stamp `ts` in the route set up request. However, it only assumes that the colluding malicious nodes have synchronized clocks. The key idea behind differential triangulation attacks is that two malicious nodes p and q can estimate the difference $\widehat{dist}(src,p) - \widehat{dist}(src,q) = t_p - t_q$ (irrespective of clock skew between (p, q) and src).

In the `candidate caller detection` step, a malicious node p computes the shortest statistical distance(s) of every node v: $dist_p[v]$. Recall that $dist_p[v]$ is a set of Pareto-optimal statistical path lengths from node v to node p. We approximate $dist_p[v]$ by computing a mean over all statistical distances $d \in dist_p[v]$ as $dist_p[v] = (\frac{1}{d} * \sum_{d \in dist_p[v]} \mu_d, \frac{1}{d} * \sum_{d \in dist_p[v]} \sigma_d^2)$. One can now compute the statistical distance $dist_{pq}[v] = dist_p[v] - dist_q[v]$ as $(\mu_p[v] - \mu_q[v], \sigma_p^2[v] + \sigma_q^2[v])$, where $dist_p[v] = (\mu_p[v], \sigma_p^2[v])$ and $dist_q[v] = (\mu_q[v], \sigma_q^2[v])$. Finally, a node v is a candidate caller if $|(\widehat{dist}(src,p) - \widehat{dist}(src,q)) - (\mu_p[v] - \mu_q[v])| < \epsilon$.

We assign a score to a node v ($score_{pq}(v)$) based on the likelihood that the statistical distance $dist_{pq}[v]$ matches the observation $t_p - t_q$. If the link latency distribution is Gaussian, then $dist_{pq}[v]$ is Gaussian as well; hence, $score_{pq}(v) = Gauss_{\mu_p[v]-\mu_q[v],\sigma_p^2[v]+\sigma_q^2[v]}(\widehat{dist}(src,p) - \widehat{dist}(src,q))$. For all other distributions, we use an approximation based on Chebyshev's inequality $score_{pq}(v) = \frac{1}{k^2}$, where $k = \frac{(\widehat{dist}(src,p) - \widehat{dist}(src,q)) - (\mu_p[v] - \mu_q[v])}{\sqrt{\sigma_p^2[v] + \sigma_q^2[v]}}$. The `triangulation` step operates as follows. An arbitrarily chosen malicious node (say p_1) is used as the reference node. We compute $score_{p_i p_1}(v)$ for all nodes v and $i > 1$. We compute the average score for a node v as $\frac{\sum_{i=2}^{n} score_{p_i p_1}(v)}{n-1}$.

D. Topology Discovery

Triangulation based timing analysis attacks require that the network topology is at least partially known. In this section we sketch techniques to discover the VoIP network topology by exploiting topology maintenance protocols. VoIP networks allow a node to discover other nodes in its neighborhood. This is typically achieved by `ping` and `pong` messages for scoped crawling. A malicious node p may send a `ping` message with a small TTL (time-to-live), say $ttl = 2$, and discover all nodes and links that are within ttl hops from node p. Additionally, the malicious node can use the time elapsed between sending a `ping` message and receiving a `pong` message from a node to estimate link latencies (mean and variance). The adversary can combine partial topology information (using a node's IP address as its identifier) from multiple malicious nodes and construct a view of the overall network topology.

E. Attacks Evaluation

To illustrate the effectiveness of our timing analysis attacks, we use a synthetic network topology with 1024 nodes. The topology is constructed using the NS-2 topology generator (GT-ITM [3], [1], [37]), and our experiments were performed on NS-2 [2]. This topology models network geography (stub domains and autonomous systems) and the small world phenomenon [15], [25].

[5] Alternatively we can compute joint distributions over all malicious nodes, but this approach does not scale

The topology generator assigns node-to-node round trip times varying from 24ms−150ms with a mean of 74ms and is within 20% error margin from real world latency measurements [19].

Deterministic Triangulation. Figure 5 shows the distribution of the number of nodes at a distance d from a randomly chosen node p. Our timing attack essentially exploits the fact that the distance distribution is not uniform and random. However, without triangulation, the attack is largely ineffective (as shown in Figure 6). Figure 6 shows the number of suspects as we vary the number of malicious nodes in the network with ϵ = 10ms. Recall that ϵ denotes an upper bound on the error in distance estimation. Observe that the number of suspects with one malicious node is 343; with 10 malicious nodes, the number of suspects is reduced drastically to 1.17 (almost uniquely identifying the caller with high probability). Compared to the shortest path triangulation attacks, multi-path triangulation attack offers only marginal benefits. For example, with five or more malicious nodes in the network, the marginal benefit of using multi-path triangulation attack over the shortest path triangulation attack is 4.3%. We observed that $dist^{\rho}(s,p)$ for $\rho > 1$ are tightly correlated with $dist^{1}(s,p)$, and are thus not good statistical discriminators. Hence, we do not consider multi-path triangulation attacks in the rest of this paper.

Figure 7 shows the number of suspects as we vary the parameter ϵ with 10 malicious nodes. When ϵ is set too small, even introducing a small uncertainty in the network link latencies (say, small jitters) may result in either an inaccurate candidate caller set. On the other hand, a large value for ϵ identifies a huge candidate caller set, thereby making the attack less effective. In the next set of experiments, we show that using statistical triangulation attack offers a better approach to capture uncertainties in network latencies.

Statistical Triangulation. Figure 8 shows that the statistical triangulation attack is more effective than a deterministic triangulation attack when there are uncertainties in link latencies. We assume that there are ten malicious nodes in the network. The figure shows the probability that the caller appears in the top-10 entries using a Gaussian distribution for link latencies (the x-axis is the link latency coefficient of variation, σ_e:μ_e). The figure shows the results for the deterministic triangulation attack using an optimal setting for the parameter ϵ. Note that initially as ϵ increases, the probability of the deterministic triangulation attack increases; however, after a critical value, increasing ϵ decreases the effectiveness of the attack.

Figure 9 shows the probability that the caller appears in the top-10 entries in the ranked list for three commonly used link latency distributions: Gaussian, Weibull and bursty models (using Pareto distribution) [2], by varying the number of malicious nodes. We set the link latency coefficient of variation, σ_e:μ_e = 0.25. Even for a small increase in the number of malicious nodes, the probability of a successful attack increases significantly. For a Gaussian distribution, we use the exact likelihood estimate; hence, the attack is more accurate. Bursty delays (typically modeled using a heavy tailed Pareto distribution) reduce the efficacy of the attacks the most because its higher order moments [26] are larger than the Weibull distribution. Figure 10 shows the probability that the caller appears in the top-κ entries in the ranked list for varying κ under three types of link latency distributions. Note that the top-1 probability is 0.25 and the top-10 probability is 0.9; this amply illustrates the attack's efficacy.

Differential Triangulation. Figure 11 compares the top-10 probability of a differential triangulation attack against a statistical triangulation attack using 10 malicious nodes. Assuming that the clocks on all nodes are synchronized, the statistical triangulation attack performs slightly better. If the clocks were synchronized, then the differential triangulation attack indeed discards useful statistical information. For instance, when the clock skew is zero, the top-10 probability in a differential triangulation attack is 0.85, while that of that of the statistical triangulation attack is 0.9. However, the efficacy of statistical triangulation attacks drop significantly as the clocks go out of synch. Figures 12 and 13 show the probability of a successful attack using three types of link latency distributions. Figure 12 shows the top-10 probability as the number of malicious proxy nodes increases for different network latency distributions; and Figure 13 shows the top-k probability using 10 malicious nodes and different network latency distributions. These results show that the differential triangulation attack can achieve a top-10 probability of 0.78 with only 10 malicious nodes even under a bursty link latency model.

Topology Discovery. Figure 14 shows the fraction of topology discovered by an adversary using the topology discovery method described in Section III-D as we vary ttl and the number of malicious nodes m. We observe that with m = 20 and ttl = 2, about 75% of the topology is discovered by the adversary. Evidently, as ttl increases, the adversary gets a more complete view of

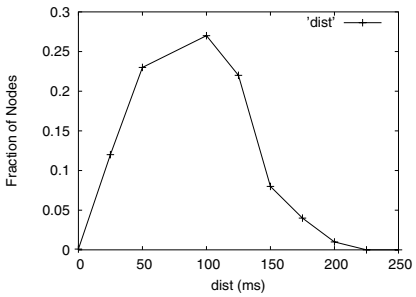

Fig. 5. Distance Distribution

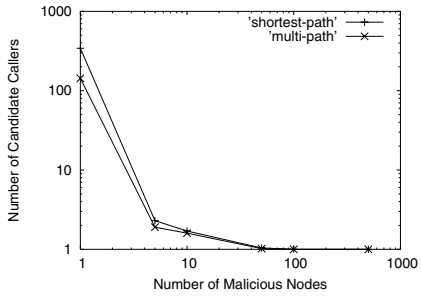

Fig. 6. Deterministic Triangulation: ϵ = 10ms

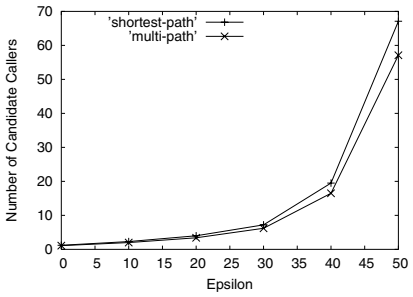

Fig. 7. Deterministic Triangulation: 10 Malicious Nodes

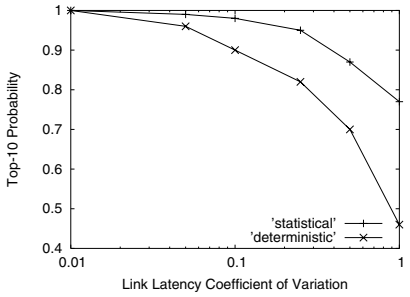

Fig. 8. Statistical Vs Deterministic Triangulation Attack

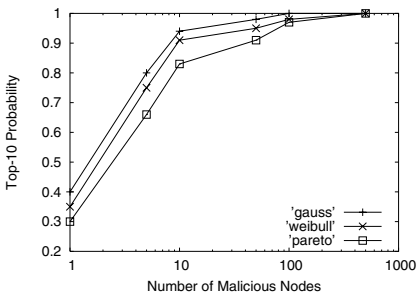

Fig. 9. Statistical Triangulation

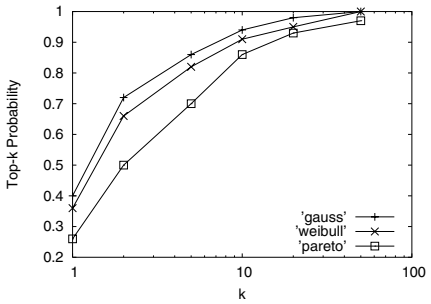

Fig. 10. Statistical Triangulation: 10 Malicious Nodes

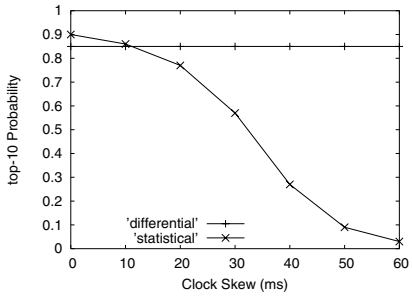

Fig. 11. Differential Vs Statistical Triangulation Attack

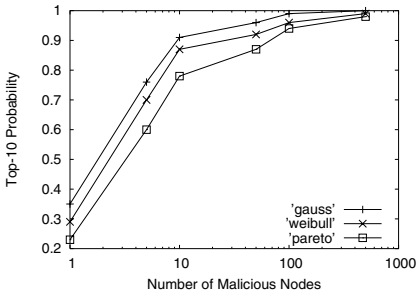

Fig. 12. Differential Triangulation: κ = 10

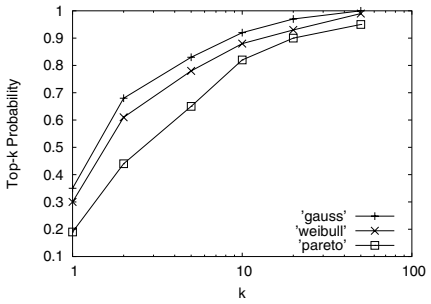

Fig. 13. Differential Triangulation: 10 Malicious Nodes

the network topology; but, using a large ttl value may allow the good nodes to *detect* topology discovery by malicious nodes. Hence, the adversary may be limited by ttl = 2, which is commonly used by topology maintenance protocols. Consequently, it may not be possible to discover the entire network topology. Additionally, the nodes may join and leave the VoIP network (churn) and hence the topology may evolve with time. To handle topology churn, the malicious nodes have to periodically *rediscover* the VoIP network in order to retain attack efficacy; but, aggressive topology rediscovery makes attack detection trivial. We evaluate the sensitivity of our attacks to incomplete knowledge of the network topology, network churn, and network size using different route set up protocols in our experimental evaluation (see Section V-C).

IV. Countering Triangulation based Timing Attacks

We have so far shown that while the broadcast based route set up protocol meets the QoS requirements (by setting up the shortest route), it is vulnerable to timing analysis attacks. This section attempts to build route set up protocols that can effectively trade off route latency with anonymity. While doing so, we exploit the fact that the route does not have to be the shortest route as long as its one-way latency is smaller than a threshold (=250ms

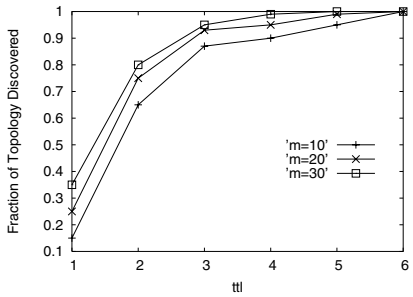

Fig. 14. Topology Discovery

for interactive voice communication). One obvious approach to countering caller identification attacks is to use *latency perturbation*; each node in the network adds a random delay before forwarding a route set up request to its neighbors in the route set up protocol. For instance, we could add Gaussian delay (μ_{noise}, σ^2_{noise}) with a low mean (for short routes) and high variance (more uncertainty into timing analysis attacks). Nonetheless, this is equivalent to replacing the latency of each edge e described by link latency (μ_e, σ^2_e) with a perturbed latency (μ_e + μ_{noise}, σ^2_e + σ^2_{noise}) and is thus susceptible to statistical and differential analysis attacks. Our experiments (see Section V) indicate that this approach does not alleviate the problem.

In this section, we demonstrate that a random walk based search algorithm is resilient to triangulation based timing attacks. However, the random walk approach may set up highly sub-optimal routes, almost blatantly violating the one-way latency constraint. We develop hybrid algorithms that combine the shortest route property of the broadcast protocol and the attack resilience property of the random walk protocol with tuneable knobs that allow us to trade off QoS guarantees with anonymity.

A. Random Walk Search Algorithm

Similar to the broadcast based route set up protocol, the random walk protocol operates in four steps.

`initSearch`. A VoIP client src initiates a search for a receiver dst by sending $search(searchId,\ sipurl = dst.sipurl)$ to a uniformly and randomly chosen neighbor $q \in ngh(src)$. The search identifier $searchId$ is a long randomly chosen unique identifier.

`processSearch`. When p receives $search(searchId,\ sipurl)$ from its neighbor q. p checks if $sipurl$ is the URL of a VoIP client connected to p. If yes, p returns its IP address $result(searchId,\ p)$ to peer q. If not, p uniformly and randomly chooses a neighbor $q \in ngh(p)$. p sends $search(searchId,\ sipurl)$ to q. If p has not previously seen the search identifier $searchId$, it caches $\langle searchId,\ sipurl,\ q\rangle$.

`processResult`. When p receives $result(searchId,\ q)$ from q. p looks up its cache of recently seen search queries to locate $\langle searchId,\ sipurl,\ prev\rangle$. p adds a routing entry $\langle sipurl,\ q\rangle$ and forwards $result(searchId,\ prev)$ to $prev$.

`finSearch`. When src receives $result(searchId,\ q)$ from q, it adds $\langle dst,\ q\rangle$ to its routing table.

There are two key properties of the random walk based search algorithm that make it resilient to triangulation based timing attacks. First, Markovian property (memoryless) of the random walk algorithm: Let us suppose that a random walker visits two malicious nodes p_1 and p_2 in time t_1 and t_2 (respectively) with $t_1 < t_2$. The Markovian property of the random walk algorithm ensures that the time instances at which the request reaches p_2 nodes do not leak any additional information to the adversary. Formally, given the fact that a random walker visited node p_1 first, the probability that it would visit node p_2 in the future is independent of the caller src.

$$Pr(visit\ p_2 \mid visit\ p_1 \wedge src = v) = Pr(visit\ p_2 \mid visit\ p_1)$$

Hence, neither the time instant t_2 nor the difference $t_2 - t_1$ provides any additional information to the adversary. In fact, the above argument holds for any number of malicious nodes visited by the random walker. Hence, when a random walker visits a set of malicious nodes p_1, p_2, $\cdots$, p_n (in that order), the only useful information is the fact that the random walker first visited node p_1. While triangulation works by combining information from multiple malicious nodes, the random walk protocol reduces the collective knowledge of n malicious nodes p_1, p_2, $\cdots$, p_n to just the first observer (say, p_1). Formally,

$$Pr(src = v \mid visit\ p_1, p_2, \cdots p_n) = Pr(src = v \mid visit\ p_1\ first)$$

Second, the random walk algorithm is not vulnerable to shortest path based timing attacks because a random walker does not essentially traverse the shortest path between any two nodes. We use the notion of *random walk distance* [7] to study the resilience of the random walk algorithm against timing analysis attacks. We define random walk distance between two nodes u and v ($rwdist_u[v]$) as the expected number of hops required for a random walker to reach node v starting from node u. Unlike the shortest path distance, the random walk distance between two nodes u and v may not be symmetric, that is, $rwdist_u[v] \neq rwdist_v[u]$.

Let p be the first malicious node that received a route set up request originating from caller src. If there exists no such node p then the search was *completely secure*. For every node v, we compute the probability a random walk starting from node v reaches malicious node p before it reaches any other malicious node in the network. Using standard Markov model theory, we model the random walk using a transition probability matrix M. An entry $M_{ij} = \frac{1}{|ngh(i)|}$ if $j \in ngh(i)$; 0 otherwise. M_{ij} denotes the probability that a random walker will visit node j given the walker is currently in node i. We determine the probability that a random walker starting from node v reaches node u in step d ($pr_v^d[u]$) from πM^d, where π is a row vector with $\pi[i] = 1$ if $i = v$; 0 otherwise. Every malicious node p computes the random walk distance from a node v as $rwdist_v[p]$:

$$rwdist_v[p] = \sum_{d=1}^{\infty} d * pr_v^d[p] * \left(\prod_{i=1}^{d-1} (1 - pr_v^i[p]) \right) \quad (1)$$

While theoretically the summation extends up to infinity, in practice $pr_v^d[p]$ tends to converge to a steady state value for all $d > N \log N$, where N is the number of nodes in the network[6]. A set of colluding malicious nodes p_1, p_2, $\cdots$, p_n where p_1 received the first random walk request operates as follows. For every node v, the adversary compiles $rwdist_v[p_i]$ for all $1 \leq i \leq n$ and sorts them in increasing order. Let us suppose that p_1 is the η^{th} smallest element in the sorted list. Then, we associate a score with node v as $score_{p_1}[v] = \eta$. We sort the nodes by their score as follows: $v_1 \prec v_2 := (score_{p_1}[v_1] < score_{p_1}[v_2]) \vee (score_{p_1}[v_1] = score_{p_1}[v_2] \wedge rwdist_{v_1}[p_1] < rwdist_{v_2}[p_1])$. Similar to the triangulation based timing attacks, the true caller is more likely to appear in the top few entries of this sorted list.

We show in our experiments sections that with 10 malicious nodes the top-10 probability for a random walk protocol is 0.05; while that for the broadcast protocol is 0.85. A major drawback with the random walk search algorithm is its unbounded route latency. In the following sections, we present hybrid techniques that attempt to combine triangulation attack resilience from the random walk search algorithm and optimal (shortest) route setup using the broadcast search algorithm.

B. Hybrid Route Set Up Protocols

1) Controlled Random Walk: In this section, we propose a controlled combination of the random walk search protocol and the broadcast search protocol. We use a global system wide parameter γ which limits the length of the random walk. In this protocol, the search algorithm operates in two phases: random walk search phase (RW) and broadcast search phase (B). The algorithm starts operating at node src in phase RW. In phase RW, when a node p receives a route set up request, it continues to operate using the random walk search algorithm with probability γ; the protocol changes to phase B with probability $1 - \gamma$. In phase B, it uses the broadcast protocol for route set up requests. Once the request enters phase B, it continues to operate in that phase.

This protocol ensures that the average number of hops in the random walk phase is $\frac{1}{1-\gamma}$ and the probability that the length of the random walk exceeds d hops is γ^d. Starting at node src, let us suppose that at the end of phase RW, the request reaches node q. The broadcast algorithm identifies the shortest path between node q to node dst. Evidently, a small value for γ ensures that the latency of the VoIP path is near optimal. A triangulation based timing attack may identify q (broadcast initiator) with high probability. However, identifying the caller src would be non-trivial to the attackers. Let the earliest malicious nodes that participated in phase RW be node p. If there exists no such malicious node, then we set p to the broadcast initiator q. Now, one can use a similar statistical inference attack described in Section IV-A to determine the caller. The key difference in the inference attack is that the parameter γ reduces the probability that a random walker starting from node v reaches node u in d hops ($pr_v^d[u]$) by a factor γ^d (in Equation 1). Clearly, a small value for γ indicates that the actual caller is close to node p and thus improves the efficacy of a statistical inference attack.

2) Multi-Agent Random Walk: The second solution is to use a multi-agent random walk search protocol. This solution is very similar to the random walk search protocol except that the caller src sends out $\omega > 1$ random walkers. As soon as the first random walker reaches the receiver dst, the route is established. Hence, sending out a large number of random walkers reduces both the route set up latency and the route latency. Indeed as ω increases, the route latency asymptotically tends to the optimal (shortest) route latency.

However, if all the random walkers were sent out by src at time $t = 0$, then this algorithm is vulnerable to triangulation based timing attacks. Let $p(rw)$ denote the first malicious node visited by random walker rw. The key idea here is that two colluding malicious nodes $p(rw_1)$ and $p(rw_2)$ can estimate $rwdist_{src}[p(rw_1)] -$

[6]Assuming M is irreducible

$rwdist_{src}[p(rw_2)]$ when they receive their first random walk request from two different random walkers rw_1 and rw_2. As discussed earlier, if the same random walker rw visits both p_1 and p_2 (in that order), then the adversary does not gain any additional information from p_2 because of the Markovian property of the random walk algorithm. However, the attacker can glean additional information, namely, $rwdist_{src}[p(rw_1)] - rwdist_{src}[p(rw_2)]$ from two different random walkers rw_1 and rw_2. Given an estimate of $rwdist_{src}[p(rw_1)] - rwdist_{src}[p(rw_2)]$, this attack is very similar to the differential triangulation attack wherein we use random walk distance $rwdist$ instead of the shortest distance $dist$ (see Section III). However, $rwdist$ has more uncertainty built into it because of the probabilistic nature of the random walk search algorithm; thus a triangulation attack on $rwdist$ is likely to be less effective than a triangulation attack on $dist$. As the number of random walkers ω increases, we may have ω malicious nodes $p(rw_i)$ $(1 \leq i \leq \omega)$ such that random walker rw_i visited malicious node $p(rw_i)$ first. Hence, a large ω increases the efficacy of the triangulation based timing attack.

V. Experimental Evaluation

In this section we present an implementation sketch followed by experimental results on various route set up protocols: broadcast (`broadcast`), latency perturbation (`lp`), random walk (`rw`), controlled random walk (`crw`) and multi-agent random walk (`marw`). All our experiments were performed on a 1024 node synthetic VoIP network topology using NS-2. Our experiments are divided into two parts. The first part measures the performance and QoS properties of these protocols. The QoS properties are intrinsically related to the parameters used: noise parameters (μ_{noise}, σ^2_{noise}) for `lp`, probability of random walk (γ) for `crw`, and number of random walkers (ω) for `marw`. We determine parameter settings for these protocols such that 99% of VoIP calls have a one-way route latency smaller than $maxLat$ = 250ms. The second part of our evaluation uses these parameter settings to evaluate the efficacy of these algorithms in defending against caller identification attacks. In the rest of this section, we assume that the attacker uses differential triangulation attacks (Section III-C) on broadcast protocol; the timing analysis on random walk protocols are as described in Section IV-B.

A. Implementation Sketch

In this section, we briefly describe an implementation of our algorithms using Phex [4]: an open source Java based implementation of peer-to-peer broadcast based route set up protocol. We have implemented our algorithms as pluggable modules that can be weaved into the Phex client code using AspectJ [14]. Our implementation is completely transparent to the VoIP protocol that operates on top of the peer-to-peer infrastructure. Below we sketch our implementation of three protocols: latency perturbation, controlled random walk and multi-agent random walk. As described in Section II, a broadcast search protocol has four parts: `initSearch`, `processSearch`, `processResult` and `finSearch`. Our algorithms require changes only to the `processSearch` part. This part is implemented in the client using several methods of which we are interested in only the following: `receiveRequest`, `checkDuplicate`, and `requestForward`.

The protocol header is modified to include our route set up protocols. Latency perturbation is implemented by intercepting an incoming route set up request delaying it for $delay \sim Gauss_{\mu_{noise},\sigma^2_{noise}}$ time units using the `receiveRequest` method. The random walk protocol requires a request to be processed even if its $searchId$ has been seen in the past; this is accomplished by bypassing the call to `duplicateCheck` method. Processing a request and setting up the route is identical for both the random walk protocols and the broadcast protocol. Finally, we need to change the request forwarding step using the `requestForward` method. In the broadcast algorithm, the `requestForward` method on node p returns all the nodes in $ngh(p)$; instead, we return only one randomly chosen neighbor. When using the controlled random walk protocol, the `requestForward` method changes the protocol header to `broadcast` with probability $1 - \gamma$.

B. Caller Identification Guards

Performance. The performance of an on-demand route set up protocol can be characterized by its messaging cost. Figure 18 shows the average messaging costs of different route set up protocols (using 100 randomly chosen pairs of callers and receivers), where N denotes the number of nodes in the network. The messaging cost for `broadcast` and `lp` is about E (E is the number of edges in the network); and that for `crw` is $\frac{1}{1-\gamma} + E$; the `rw` and `marw` protocols incur a average messaging cost of $N \log N$. While the number of edges E can be as large as $\frac{N(N-1)}{2}$, in an Internet-like topology the number of edges E is roughly βN, where β is a constant between 2 and 3. Fortunately, route set up requests are significantly smaller than the amount of data transferred in the voice

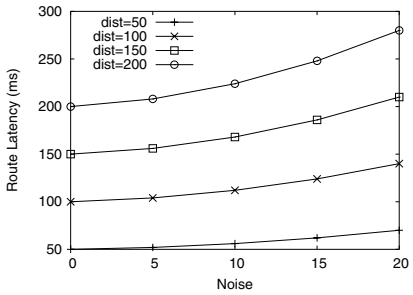

Fig. 15. Latency Perturbation

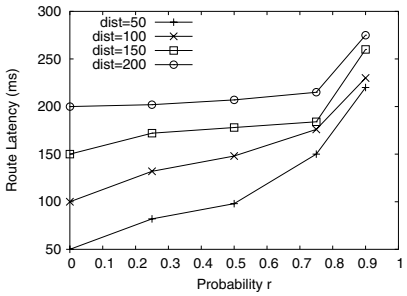

Fig. 16. Controlled Random Walk

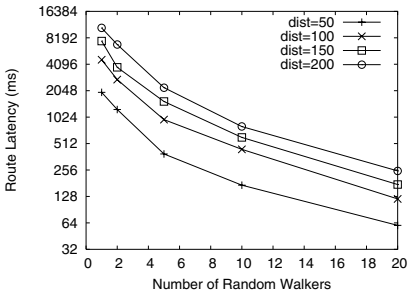

Fig. 17. Multi-Agent Random Walk

N	broadcast	lp	crw: γ = 0.75	crw: γ = 0.9	rw	marw: ω = 10	marw: ω = 20
256	563	563	567	573	945	1001	1036
1024	2560	2560	2564	2570	5089	5213	5287

Fig. 18. Messaging Cost

	99%	95%	90%
σ_{noise}	14	15	20
γ	0.76	0.79	0.83
ω	20	18	12

Fig. 19. Optimal Parameter Setting

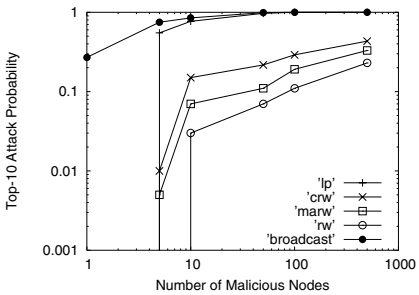

Fig. 20. Top-10 Probability

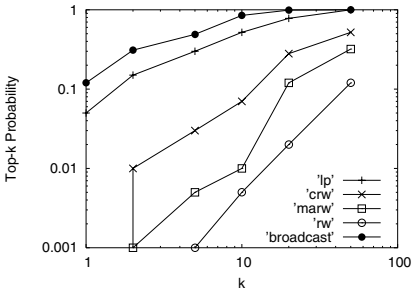

Fig. 21. Top-k Probability: 10 malicious nodes

session; hence, an increase in messaging cost by a factor $\frac{\log N}{\beta}$ (in the rw and marw protocols) does not cause discernible changes to VoIP's aggregate messaging cost.

QoS Guarantees. The QoS property of an on-demand route set up protocol can be characterized by route latency and route set up latency. Figures 15, 16 and 17 show the route latency under various parameter settings. Note that all routes with latency smaller than 250ms satisfy the QoS requirement. In the figures, 'dist' denotes the latency of the shortest path between the caller and the receiver in milliseconds. Route set up latency determines the time period between a caller initiating a route set up request and the establishment of a route between the caller and the receiver. Larger route set up latency causes an initial delay in session set up but does not affect the quality of the voice conversation. We observed that the route set up latency seldom exceeded the route latency by 20%.

Optimal Parameter Settings. We randomly chose 1024 pairs of callers and receivers. We vary algorithm parameters: increase σ^2_{noise}, γ and decrease w until $X\%$ of the pairs have a route latency under $maxLat$. We use a binary search strategy to identify the optimal parameter values. For instance, we determine σ^2_{noise} that satisfies the $X\%$ latency constraint as follows. We start with a range $(0, 100)$, where σ_{noise} = 0 satisfies the $X\%$ latency constraint, while σ_{noise} = 100 does not satisfy the constraint. Given a range (l, u), we experiment with σ_{noise} set to $\frac{l+u}{2}$. If σ_{noise} = $\frac{l+u}{2}$ satisfies the $X\%$ latency constraint, then the new range is set to $(\frac{l+u}{2}, u)$; otherwise the new range is set to $(l, \frac{l+u}{2})$. We repeat this binary search until the size of the range $(u - l)$ is acceptably small. When the search terminates, we have the optimal parameter setting σ^{opt}_{noise} = l. Figure 19 summarizes our parameter settings for different percentile latency constraints. Our goal is to study the resilience of these algorithms under the constraint that the route latency is smaller than $maxLat$. Note that the resilience to caller identification attacks monotonically increases with σ^2_{noise} and γ and monotonically decreases with ω. Hence, studying attack resilience under the parameter

settings in figure 19 is sufficient.

Attack Resilience. In this section, we compare the attack resilience of our route set up protocols. Giving at most importance to the quality of the voice session, we use the 99% optimal parameter setting from figure 19 in this experiment. We use ω = 1 random walker (`rw`) and the broadcast based shortest route set up protocol for base line comparison. However, one should keep in mind that the random walk search algorithm (`rw`) violates the latency constraint. Figure 20 shows the top-10 probability using different route set up protocols. Figure 21 shows the top-κ probability with 10 malicious nodes using different route set up protocols. These figures indicate that the latency perturbation does not offer much resilience to caller identification attacks. The multi-agent random walk algorithm performs the best achieving a top-10 probability of 0.075 with 10 malicious nodes. Note that this is higher than the ω = 1 random walk algorithm which achieves a top-10 probability of 0.05 with 10 malicious nodes. Also, the random walk based protocols present a significant improvement over the broadcast protocol which achieves a top-10 probability of 0.85 with 10 malicious nodes.

C. Network Topology

In this section we evaluate the sensitivity of timing analysis attacks to the network topology. We show that a triangulation based timing analysis attack remains effective on a broadcast protocol even when part of the network topology is unknown (or incorrectly known) to the adversary. We also study the scalability of caller identification attacks on various route set up protocols.

Topology Discovery. Figure 22 shows the top-10 probability using different route set up protocols when only a fraction of the network topology has been discovered by the adversary. If the adversary's view of the network topology is not connected, then we simply use the first connected component that receives the route set up request for the attack. From figures 14 and 22, we observe that in a broadcast based route set up protocol, using m = 20 and ttl = 2 improves top-10 probability to 0.72. Hence, even with an incomplete knowledge of the network topology, the attacker can successfully launch caller identification attacks on a broadcast based route set up protocol. The top-10 probability for the controlled random walk and multi-agent random walk are considerably lower at 0.08 and 0.05 respectively. The random walk based protocols are more sensitive to topology knowledge and are thus more resilient to caller identification attacks. Using the multi-agent random walk protocol, the top-10 probability is 42% below that of complete topology knowledge; while the numbers for controlled random walk and the broadcast protocol are about 33% and 9% respectively.

Network Churn. Figure 23 shows the efficacy of caller identification attacks as the network churn increases for m = 10 malicious nodes. For instance, $churn$ = 0.1 represents the fact that about 10% of the proxy nodes have joined and left the network since the last discovery phase. We observe that for the broadcast protocol, the attack efficacy drops by less than 15% when 5-10% of the topology has changed since the last crawl. In contrast, for the random walk based protocols, the attack efficacy drops by 40% when 5-10% of the topology changes. Given that the average lifetime of a VoIP proxy is 10.5 hours [34], one can show that the average time required for 10% churn is one hour. More time allows a stealthy attack wherein the adversary does not frequently flood the network with topology discovery (`ping`) messages.

Network Size. Figure 24 shows the number of malicious node required to achieve a top-10 probability of 0.1 as the VoIP network size increases (x-axis is in log scale); `broadcast` nearly coincides with the x-axis. Experiments on topologies whose size was larger than 1024 nodes were performed using a discrete event simulation; the results obtained from our discrete event simulator were within 5% of that of NS-2 for topologies of size up to 1024 nodes. We observe the number of malicious nodes required for an attack grows very slowly with the size of the network for the broadcast based route set up protocol. For instance, using a 10,000 node network topology, the attacker could achieve top-10 probability of 0.1 with just 25 malicious nodes using the broadcast protocol; while it required 400 nodes in the case of controlled random walk protocol and 1200 nodes in the case of multi-agent random walk. This clearly demonstrates the ability of random walk based protocols to limit the scalability of caller identification attacks.

VI. Related Work

Several low latency anonymizing networks have been studied in the literature [12], [17], [8]. Anonymizing networks protect the identity of the sender and receiver using the concept of a mix [10], [28]. Onion routing [17] and its second generation Tor [12] aim at providing anonymous transport of TCP flows over the Internet. Tor supports on-demand route set up, but it is not QoS sensitive. ISDN mixes [23] propose solutions to anonymize

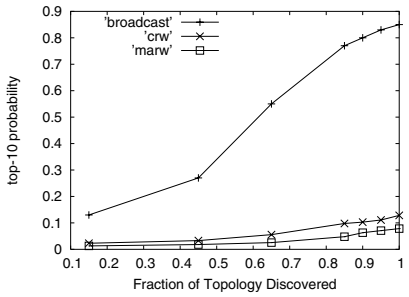

Fig. 22. Partial Topology

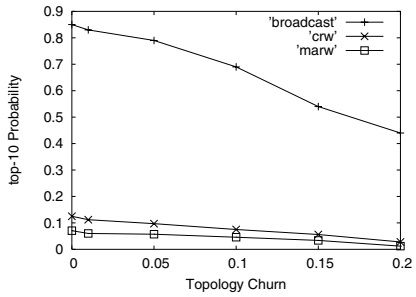

Fig. 23. Network Churn

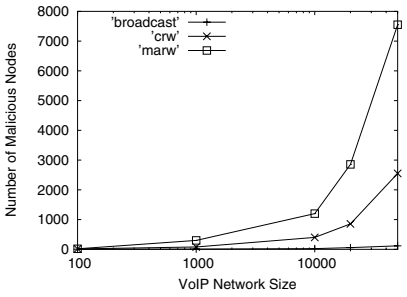

Fig. 24. Scalability

phone calls over traditional PSTN (Public Switched Telephone Networks).

It is widely acknowledged that low latency anonymizing networks are vulnerable to timing analysis attacks [30], especially from well placed malicious attackers [35]. Several papers have addressed the problem of tracing encrypted traffic using timing analysis [33], [36], [38], [32], [9], [13], [31]. All these papers use inter-packet timing characteristics for tracing traffic. The timing characteristic may be obtained either by passively observing the traffic or by actively embedding a timing signature in the traffic. Our approach differs from the above approaches in several ways. We do not rely on inter-packet timing characteristics in the data transfer phase. Our approach specifically targets QoS sensitive route set up protocols. Our timing analysis attack is closer to RF localization techniques that break source location privacy in sensor networks [18], [20]. Phantom routing [20] also proposes the use of random-walk based protocols to mitigate localization attacks in sensor networks.

Some authors have suggested that the VoIP network can be doubled to act as an anonymizing network [31]. However, the notion of setting up a *circuit* or a *route* on a VoIP network is different from that on state of art anonymizing networks in the following respects. First, a random route on an anonymizing network may not meet latency-based QoS requirements. Hence, VoIP uses route set up protocols that identify the *shortest* route from a caller to a receiver. Second, routes (or circuits) in a VoIP network are not constructed a priori. VoIP supports client mobility by allowing a client to connect to *any* node in the proxy network[7]. Hence, VoIP applications use an on-demand QoS sensitive route set up protocol.

Tarzan [16] presents an anonymizing network layer using the peer-to-peer model. It uses a gossip based approach to disseminate the identity of nodes on the peer-to-peer network. At the end of the gossip protocol, the caller src supposedly knows the network topology and the receiver dst, thus obviating the need for a route set up protocol. This approach requires the caller and receiver to be connected to the network for enough time before they can discover each other (and make VoIP calls). However, most VoIP applications use on-demand route set up to support client mobility.

Traditionally, multicast and broadcast protocols have been used to protect receiver anonymity [24]. This approach sends a unicast message to a single destination by multicasting the message to a group containing the destination. The intended destination recognizes the message is intended for itself because it expects this message (Hordes [27]) or because the sender addresses the message implicitly [24]. In contrast to these approaches, our paper focuses on caller (source) anonymity as against receiver anonymity. Perng et. al. [22] have shown that multicasting data on an anonymizing network can break some privacy guarantees. In particular, they show that the malicious nodes in the network can use the popularity of a multicast packet to infer some information about the data.

VII. Conclusion

In this paper we have addressed the problem of on-demand construction of QoS sensitive routes on anonymizing networks. We have used VoIP as a sample application to demonstrate potential attacks on QoS sensitive route set up protocols and proposed solutions to mitigate them. First, we have identified triangulation based timing attacks on broadcast based route set up protocols that can identify the caller with high probability. Second, we have developed random walk based protocols to mitigate this attack while suitably trading off QoS guarantees with anonymity. We have described a prototype implementation of our guards and presented a detailed experimental evaluation that demonstrates its QoS properties and

[7]Although clients typically connect to the *nearest* proxy network node

attack resilience. We have shown while the broadcast based route set up protocol is vulnerable to timing analysis attacks, the random walk based protocols can offer good attack resilience while satisfying QoS guarantees. In summary, our solutions effectively protect anonymity for callers in a VoIP network.

Acknowledgement: This research is partially supported by NSF CSR and CyberTrust.

REFERENCES

[1] GT-ITM: Georgia tech internetwork topology models. http://www.cc.gatech.edu/projects/gtitm/.

[2] The network simulator NS-2. http://www.isi.edu/nsnam/ns/.

[3] The network simulator NS-2: Topology generation. http://www.isi.edu/nsnam/ns/ns-topogen.html.

[4] Phex client. http://www.phex.org.

[5] Skype - the global internet telephone company. http://www.skype.com.

[6] Telegeography research. http://www.telegeography.com.

[7] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs. http://stat-www.berkeley.edu/users/aldous/RWG/book.html.

[8] A. Back, I. Goldberg, and A. Shostack. Freedom 2.1 security issues and analysis. Zero Knowledge Systems, Inc. white paper, 2001.

[9] A. Blum, D. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In *7th RAID*, 2004.

[10] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of ACM, 24(2): 84-88*, 1981.

[11] L. Deng and M. D. F. Wong. An exact algorithm for statistical shortest path problem. In *Design Automation Conference*, 2006.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second generation onion router. In *13th USENIX Security Symposium*, 2000.

[13] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. Multiscale stepping stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In *5th RAID*, 2002.

[14] Eclipse. Aspectj compiler. http://eclipse.org/aspectj.

[15] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. In *IEEE Journal on Special Areas in Communication*, 2002.

[16] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM CCS*, 2002.

[17] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. In *Communications of ACM, Vol 42(2)*, 1999.

[18] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *ACM MobiSys*, 2003.

[19] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *ACM SIGCOMM*, 2003.

[20] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source location privacy in sensor network routing. In *IEEE ICDCS*, 2005.

[21] A. Papoulis. Probability, random variables, and stochastic processes. In *3rd ed. McGraw Hill ISBN 0-07-100870-5, pp. 113-114*.

[22] G. Perng, M. K. Reiter, and C. Wang. M2: Multicasting mixes for efficient and anonymous communication. In *IEEE ICDCS*, 2006.

[23] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-MIXes: Untraceable communication with small bandwidth overhead. In *GI/ITG Conference on Communication in Distributed Systems*, 1991.

[24] A. Pfitzmann and M. Waidner. Networks without user observability. In *Computers and Security, 2(6): 158-166*, 1987.

[25] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *12th IEEE INFOCOM*, 2001.

[26] S. M. Ross. Introduction to probability models. Academic Press ISBN: 0125980558, 2002.

[27] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *ACM CCS*, 2000.

[28] V. Shmatikov and M. H. Wang. Timing analysis in low latency mix networks: Attacks and defenses. In *11th ESORICS*, 2006.

[29] G. I. Sound. VoIP: Better than PSTN? http://www.globalipsound.com/demo/tutorial.php.

[30] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[31] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the internet. In *12th ACM CCS*, 2005.

[32] X. Wang and D. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *10th ACM CCS*, 2003.

[33] X. Wang, D. Reeves, and S. Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *7th ESORICS*, 2002.

[34] X. Wang, Z. Yao, and D. Loguinov. Residual-based measurement of peer and link lifetimes in gnutella networks. In *IEEE InfoCom*, 2007.

[35] X. Y. Wang and D. S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *ACM CCS*, 2003.

[36] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. In *6th ESORICS*, 2000.

[37] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork? In *IEEE Infocom*, 1996.

[38] Y. Zhang and V. Paxon. Detecting stepping stones. In *9th USENIX Security Symposium*, 2000.