Privacy in VoIP Networks: Flow Analysis Attacks and Defense

Mudhakar Srivatsa[†], Arun Iyengar[†], Ling Liu[‡] and Hongbo Jiang[*]
IBM T.J. Watson Research Center[†]
College of Computing, Georgia Institute of Technology[‡]
Huazhong Institute of Science and Technology[*]
{msrivats, aruni}@us.ibm.com, lingliu@cc.gatech.edu, hongbojiang@hust.edu.cn

***Abstract*— [1] Peer-to-Peer VoIP (voice over IP) networks, exemplified by Skype [5], are becoming increasingly popular due to their significant cost advantage and richer call forwarding features than traditional public switched telephone networks. One of the most important features of a VoIP network is privacy (for VoIP clients). Unfortunately, most peer-to-peer VoIP networks neither provide personalization nor guarantee a quantifiable privacy level. In this paper we propose novel flow analysis attacks that demonstrate the vulnerabilities of peer-to-peer VoIP networks to privacy attacks. We then address two important challenges in designing privacy-aware VoIP networks: Can we provide personalized privacy guarantees for VoIP clients that allow them to select privacy requirements on a per-call basis? How to design VoIP protocols to support customizable privacy guarantee? This paper proposes practical solutions to address these challenges using a quantifiable k-anonymity metric and a privacy-aware VoIP route setup and route maintenance protocols. We present detailed experimental evaluation that demonstrates the performance and scalability of our protocol, while meeting customizable privacy guarantees.**

I. Introduction

The concept of a mix [9] was introduced by Chaum in 1981. Since then several authors have used mix as a network routing element to construct anonymizing networks such as Onion Routing [16], Tor [10], Tarzan [15], or Freedom [7]. Mix network provides good anonymity for high latency communications by routing network traffic through a number of nodes with *random delay* and *random routes*. However, emerging applications such as VoIP, SSH, online gaming, etc have additional quality of service (QoS) requirements; for instance ITU (International Telecommunication Union) recommends up to 250ms one-way latency for interactive voice communication; recent case study [26] indicates that latencies up to 250ms are unperceivable to human users; while latencies over 400ms significantly deteriorate the quality of voice conversations.

This paper examines anonymity for QoS sensitive applications on mix networks using peer-to-peer VoIP service as a sample application. A peer-to-peer VoIP network typically consists of a core proxy network and a set of clients that connect to the edge of this proxy network (see Fig 1). This network allows a client to dynamically connect to any proxy in the network and to place voice calls to other clients on the network. VoIP uses the two main protocols: route setup protocol for call setup and termination, and Real-time Transport Protocol (RTP) for media delivery. In order to satisfy QoS requirements, a common solution used in peer-to-peer VoIP networks is to use a route setup protocol that sets up the *shortest route* on the VoIP network from a caller src to a receiver dst[2]. RTP is used to carry voice traffic between the caller and the receiver along an established bi-directional voice circuit.

In such VoIP networks, preserving the anonymity of caller-receiver pairs becomes a challenging problem. In this paper we focus on attacks that attempt to infer the receiver for a given VoIP call using traffic analysis on the media delivery phase. We make two important contributions. First, we show that using the shortest route (as against a random route) for routing voice flows makes the anonymizing network vulnerable to *flow analysis attacks*. Second, we develop practical techniques to achieve quantifiable and customizable k-anonymity on VoIP networks. Our proposal exploits the fact that audio codecs (such as G.729A without silence suppression[3]) generate *statistically identical* packet streams that can be mixed without leaking much information to an external observer (see Fig 2).

The following portions of this paper are organized as follows. We present a reference model for a VoIP network followed by flow analysis attacks in Section III. Section IV provides a more concrete definition of k-anonymity and describes an efficient anonymity-aware route setup protocol (AARSP). We sketch an implementation of our proposal and present experimental results that quantify the performance and scalability of AARSP in Section V. We present related work in Section VI and finally conclude in Section VII.

II. VoIP Route Setup Protocol

In this section, we describe a commonly used shortest route setup protocol in peer-to-peer VoIP networks. The protocol operates in four steps: `initSearch` (initiates a route setup by src), `processSearch` (process route setup request at some

[1] A short version of this paper appears in IEEE INFOCOM 2009: http://www.research.ibm.com/people/i/iyengar/INFOCOM2009-kanon.pdf.
Prof. Ling Liu's work was partially supported by grants from NSF CyberTrust program, AFOSR, Intel, and an IBM faculty award

[2] Enterprise VoIP networks that use SIP or H.323 signaling protocol may not use the shortest route

[3] G.729A without silence suppression deterministically generates one IP packet every 20ms. With silence suppression voice flows may be non-identical, thereby making them trivially vulnerable to traditional traffic analysis attacks

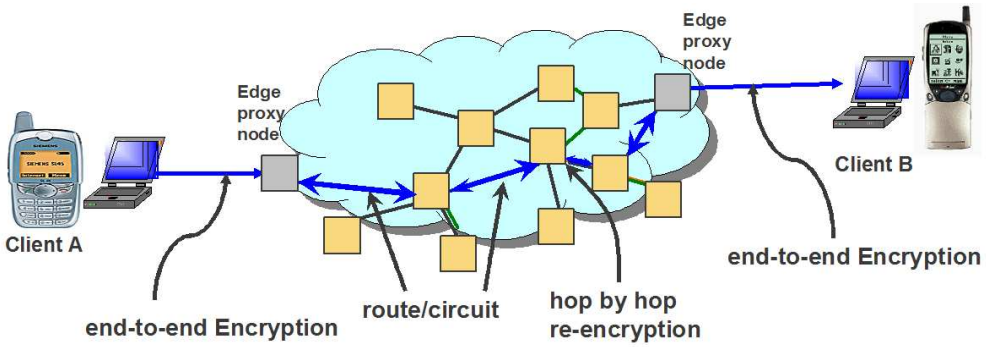

Fig. 1. Anonymizing VoIP Network

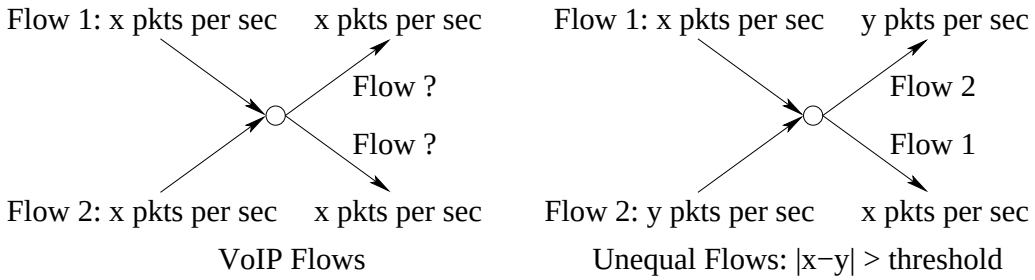

Fig. 2. Mixing Statistically Identical VoIP Flows

node), `processResult` (process results of a route setup request at some node), and `finSearch` (concludes the route setup). One should note that flow analysis attacks exploits only the shortest path property and is *independent* of the concrete route setup protocol. The description here serves as a basis for our AARSP in Section IV.

`initSearch`. A VoIP client src initiates a route setup 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. Each VoIP client is identified by an URL (say, `sip:bob@example.com`). The search identifier $searchId$ is a long randomly chosen unique identifier and ts denotes the time stamp at which the search 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 search 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. p broadcasts $search(searchId,\ sipurl,\ ts)$ to all $p' \in ngh(p) - \{q\}$ and caches the search identifier $\langle searchId,\ sipurl,\ q\rangle$ in its recently seen list. Note that p' has no knowledge of where the search 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.

The route setup protocol establishes the shortest overlay network route between src and dst. This observation follows from the following facts: (i) the first search request that reaches a node p must have traveled along the *shortest* route from src to p, and (ii) In `processSearch` a node p records the neighbor q through which it received the first search request. This indicates that the shortest route from src to p is via q. Setting $p = dst$ shows that route setup procedure in `processResult` builds the shortest VoIP network path from src to dst.

After a successful route setup, the clients src and dst exchange an end-to-end media encryption key and switch to the media delivery phase. The media delivery phase additional uses hop-by-hop re-encryption using pair-wise shared keys between neighboring proxy nodes in the VoIP network. An external observer tapping into the VoIP network may observe $\langle srcIP, dstIP, srcPort, dstPort, E_{K_{p,q}}(E_{K_{src,dst}}(media))\rangle$, where $K_{p,q}$ denotes a pair-wise shared symmetric key between neighboring nodes p and q on the route, $K_{src,dst}$ denotes the end-to-end encryption key and $media$ denotes an encoding of media bits. Further, an observer may observe packet sizes[4] and statistics on inter-packet arrival times. Re-encryption essentially guarantees unlinkability between $E_{K_{p,q}}(E_{K_{src,dst}}(media))$ and $E_{K_{q,r}}(E_{K_{src,dst}}(media))$ by examining the contents of the transmitted packet.

III. Flow Analysis Attacks

In this section, we describe flow analysis attacks on VoIP networks. These attacks exploit the shortest path nature of the voice flows to identify pairs of callers and receivers on the VoIP network. Similar to other security models for VoIP networks, we assume that the physical network infrastructure is owned by an untrusted third party (say, tier one/two network

[4]Media packet sizes obtained from a common encoding algorithm such as G.729A are typically identical; if not packets can be padded with random bits

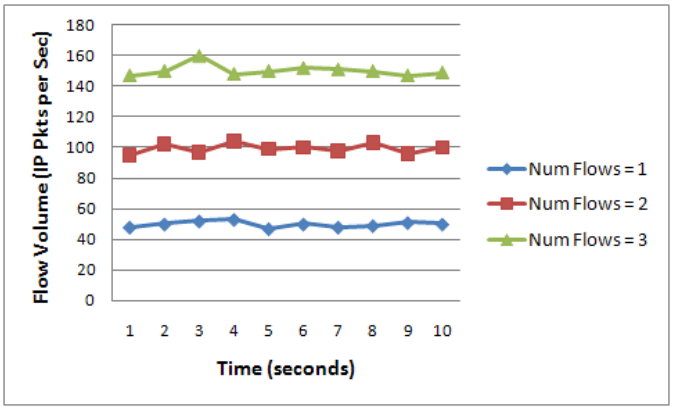

Fig. 3. Inferring Number of Flows from Flow Volume

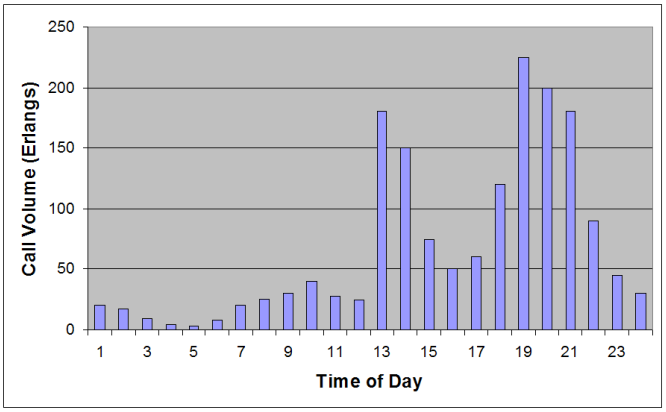

Fig. 4. Call Volume Data

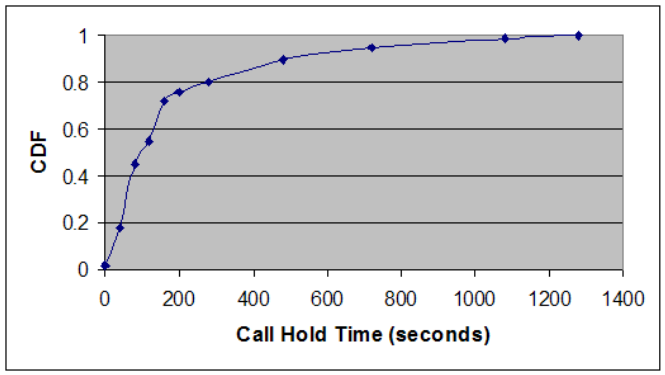

Fig. 5. Call Hold Time Data

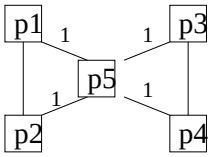

Fig. 6. Tracing Voice Calls

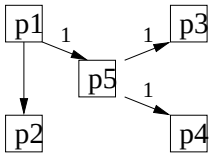

Fig. 7. Shortest Path Tracing

service provider). Hence, the VoIP service must route voice flows on the untrusted network in a way that preserves the identities of callers and receivers from the untrusted network. We assume that the untrusted network service provider (*adversary*) is aware of the VoIP network topology [28][10] and the flow rates on *all* links in the VoIP network [19][7]. The network service provider can obtain VoIP topology and flow information using traffic analysis (see Fig 3) or using various measurement based approaches (such as expanding ring search on the network topology) [23]. We experimentally show that the attack can be very effective even when only one third of the links are monitored by the adversary.

We represent the VoIP network topology as a weighted graph $G = \langle V, E \rangle$, where V is the set of nodes and $E \subseteq V \times V$ is the set of undirected edges. The weight of an edge $e = (p, q)$ (denoted by $w(p, q)$) is the latency between the nodes p and q. We assume that the adversary can observe the network and thus knows $nf(p \rightarrow q)$ the number of voice flows between two nodes p and q on the VoIP network such that $(p, q) \in E$.

To illustrate the effectiveness of our flow analysis attacks, we use a synthetic network topology with 1024 nodes. The topology is constructed using the GT-ITM topology generator [35][1] and our experiments were performed on NS-2 [2][3]. GT-ITM models network geography and the small world phenomenon (power law graph with parameter γ=2.1) [14][22]. 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 [17]. The average route (shortest path) latency between any two nodes in the network is 170ms, while the worst case route latency is 225ms. Our experiments over NS-2 use a bursty packet delay model wherein 20% of the packets incur an additional delay of up to 44% of average one-way latency [17]. In practice, the total cost of framing, decoding and hop-by-hop re-encryption amounts to about 1.4ms per voice packet on commodity hardware. In our simulations, we adjust link latencies to reflect the cost of routing VoIP packets.

We generate voice traffic based on call volume and call hold time distribution obtained from a large enterprise with 973 subscribers (averaged over a month) (see Figures 4 and 5). The call volume is specified in Erlangs [18]: if the mean arrival rate of new calls is λ per unit time and the mean call holding time (duration of voice session) is h, then the traffic in Erlangs $A = \lambda h$; for example, if total phone use in a given area per hour is 180 minutes, this represents 180/60 = 3 Erlangs. We use G.729A audio codec for generating audio traffic. The (src, dst) pair information for each call was not made available. We have experimented under two settings: first, we assume that for a given VoIP call, the (src, dst) pair is chosen randomly from the VoIP network; second, we assume that 80% of the calls are made between nodes that are in the same network geography (e.g., same autonomous system). As noted in Section III-E, any prior information (such as, 80% of call volume is limited to local network geography) can be used by the adversary to further enhance the efficacy of flow analysis attacks. Finally, we note that all results reported in this paper have been averaged over seven independent runs.

A. Naive Tracing Algorithm

Let src be the caller. We use a Boolean variable $f(p) \in \{0, 1\}$ to denote whether the node p is reachable from src using

```
TRACE(Graph G=⟨V, E⟩, Caller src)
(1)   for each vertex v ∈ V
(2)       f[v] = 0; label[v] = false
(3)   end for
(4)   f[src] = 1; label[src] = true
(5)   while pick a vertex v labeled true
(6)       label[v] = false
(7)       for each node u such that (u, v) ∈ E
(8)           if (f[u] = 0)
(9)               f[u] = 1; label[u] = true
(10)          end if
(11)      end for
(12)  end while
```

Fig. 8. Naive Tracing Algorithm

the measured flows on the VoIP network. One can determine $f(p)$ for all nodes p in $O(|E|)$ time as follows. The base case of the recursion is $f(src) = 1$. For any node q, we set $f(q)$ to one if there exists a node p such that $(p, q) \in E \wedge f(p) = 1 \wedge nf(p \rightarrow q) > 0$.

Let us consider a sample topology shown in Figure 6. For the sake of simplicity assume that each edge has unit latency. The label on the edges in Figure 6 indicates the number of voice flows. A trace starting from caller p_1 will result in $f(p_1) = f(p_2) = f(p_3) = f(p_4) = f(p_5) = 1$. Filtering out the VoIP proxy nodes (p_5) and the caller (p_1), the clients p_2, p_3 and p_4 could be potential destinations for a call emerging from p_1.

However, the tracing algorithm does not consider the shortest path nature of voice routes. Considering the shortest path nature of voice paths leads us to conclude that p_2 is not a possible receiver for a call from p_1. If indeed p_2 were the receiver then the voice flow would have taken the shorter route $p_1 \rightarrow p_2$ (latency = 1), rather than the longer route $p_1 \rightarrow p_5 \rightarrow p_2$ (latency = 2) as indicated by the flow information. Hence, we now have only two possible receivers, namely, p_3 and p_4.

B. Shortest Path Tracing Algorithm

In this section, we describe techniques to generate a directed sub-graph $G^1 = \langle V^1, E^1 \rangle$ from G which *encodes* the shortest path nature of the voice paths. Given a graph G and a caller src, we construct a sub-graph G^1 that contains only those voice paths that respect the shortest path property. Figure 9 uses a breadth first search on G to compute G^1 in $O(|E|)$ time.

One can formally show that the directed graph G^1 satisfies the following properties: (i) If the voice traffic from src were to traverse an edge $e \notin E^1$, then it violates the shortest path property. (ii) All voice paths that respect the shortest path property are included in G^1. (iii) The graph G^1 is acyclic.

Figure 7 illustrates the result of applying the algorithm in Figure 9 on the sample topology in Figure 6. Indeed if one uses the trace algorithm (Figure 8) on graph G^1, we get $f(p_2) = 0$, $f(p_3) = f(p_4) = 1$. Figure 10 compares the effectiveness of the shortest path tracing algorithm with the tracing algorithm on a 1024 node VoIP network. On the x-axis we plot the call traffic measured in Erlangs. We quantify the efficacy of an attack using standard metrics from inference algorithms: *precision*, *recall* and *F-measure*. We use S to denote the set

```
SHORTEST PATH TRACING(Graph G=⟨V, E⟩, Caller src)
(1)   for each vertex v ∈ V
(2)       dist[v] = ∞; label[v] = false; prev[v] = null
(3)   end for
(4)   dist[src] = 0; label[src] = true
(5)   while pick a labeled vertex v with minimum dist[v]
(6)       label[v] = false
(7)       for each node u such that (u, v) ∈ E
(8)           if (dist[u] < dist[v] + w(u, v))
(9)               dist[u] = dist[v] + w(u, v)
(10)              prev[u] = {v}; label[u] = true
(11)          end if
(12)          if (dist[u] = dist[v] + w(u, v))
(13)              prev[u] = prev[u] ∪ {v}
(14)          end if
(15)      end for
(16)  end while
(17)  G^1 = ⟨V^1, E^1⟩: V^1 = V, E^1 = (u → v) ∀u ∈ prev[v], ∀v ∈ V
```

Fig. 9. Shortest Path Tracing Algorithm

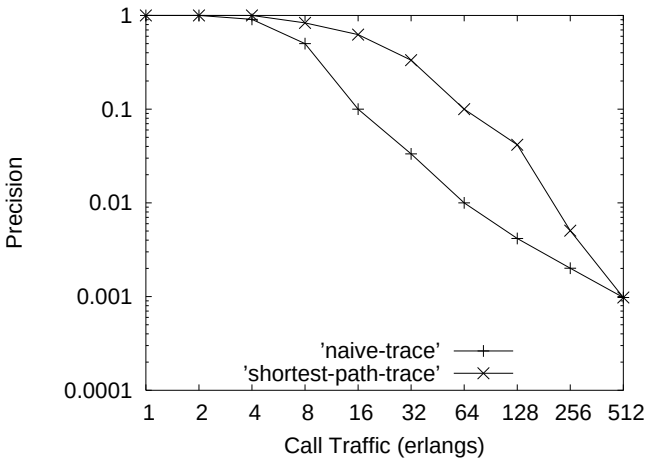

Fig. 10. Shortest Path Tracing Vs Naive Tracing

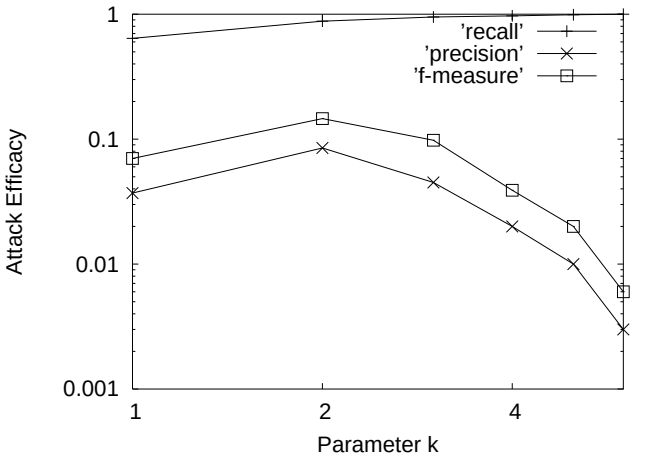

Fig. 11. Statistical Shortest Path Tracing: 128 Erlang Call Volume

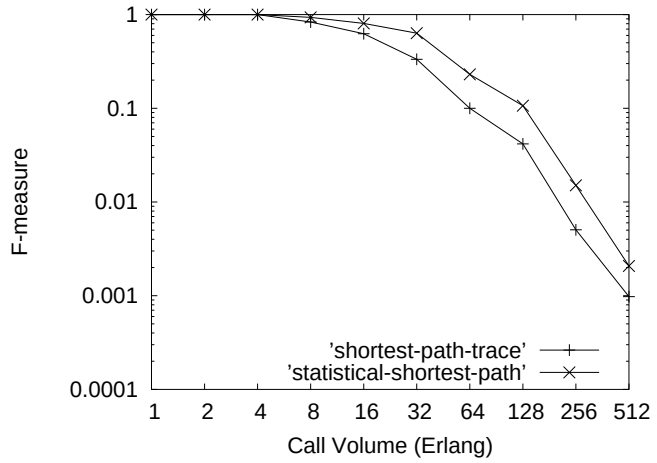

Fig. 12. F-measure with κ = 2

of nodes such that for every $p \in S$, $f[p]$ = 1. *Recall* denotes the probability of identifying the true receiver dst ($dst \in S$?), and *precision* is inversely related to the size of candidate receiver set ($\propto \frac{1}{|S|}$). F-measure (computed as harmonic mean of recall and precision scores) is a commonly used as a single metric for measuring the effectiveness of inference algorithms [29].

$$\begin{aligned} recall &= Pr(f[dst] = 1) \\ precision &= \begin{cases} \frac{1}{|S|} & \text{if } dst \in S \\ 0 & \text{otherwise} \end{cases} \\ \textit{F-measure} &= \frac{2 * recall * precision}{recall + precision} \end{aligned}$$

In a deterministic network setting, the receiver dst is guaranteed to be marked with $f[dst]$ = 1, that is, $recall$ = 1 for both the naive tracking algorithm and the shortest path tracing algorithm. Hence, Figure 10 compares only the precision of these two algorithms. We observe that for low call volumes ($<$ 64 Erlangs) the shortest path tracing algorithm is about 5-10 times more precise than the naive tracking algorithm. However, higher call volumes facilitate natural mixing of VoIP flows thereby decreasing the precision of both the naive tracing and shortest path tracing algorithms.

C. Statistical Shortest Path Tracing

In a realistic setting with uncertainties in network latencies the shortest path tracing algorithm may not identify the receiver. We handle such uncertainties in network link latencies by using a top-κ shortest path algorithm to construct G^κ from G. An edge e is in G^κ if and only if it appears in some top-κ shortest path originating from src in graph G. We modified Algorithm 9 to construct G^κ by simply maintaining top-κ distance measurements $dist^1[v]$, $dist^2[v]$, $\cdots$, $dist^\kappa[v]$ instead of only the shortest (top-1) distance measurement. We also maintain previous hops $prev^1[v]$, $prev^2[v]$, $\cdots$, $prev^\kappa[v]$ that correspond to each of these top-κ shortest paths. We add an edge (u, v) to E^κ if $u = prev^i[v]$ for some $1 \leq i \leq \kappa$. We say that the voice traffic from src to v satisfies the top-κ shortest path property if it is routed along one of the top-κ shortest paths from src to v. One can formally show that all voice paths that respect the top-κ shortest path property are included in G^κ. However, unlike G^1, the graph G^κ (for $\kappa \geq 2$) may contain directed cycles.

Evidently, as κ increases, the tracing algorithm can accommodate higher uncertainty in network latencies, thereby improving *recall*. On the other hand, as κ increases, the *precision* initially increases and then decreases. The initial increase is attributed to the fact when κ is small the tracing algorithm may even fail to identify the actual receiver as a candidate receiver; $f[dst]$ may be 0 resulting in zero precision. However, for large values of κ, the number of candidate receivers becomes very large, thereby decreasing the *precision* metric. Figure 11 shows the precision, recall and F-measure of the statistical shortest path tracing algorithm with 128 Erlang call volume and varying κ. This experiment leads us to conclude that κ = 2 yields a concise and yet precise list of potential receivers; observe that κ = 2 improves precision and recall by 97% and 37.5% (respectively) over κ = 1.

Figure 12 compares the F-measure for the statistical shortest path tracing algorithm (κ = 2) and the shortest path tracing algorithm (κ = 1) and varying call volume. We observe that for low call volumes the shortest path tracing algorithm is sufficiently accurate. However, for moderate call volumes the statistical shortest path tracing algorithm can improve attack efficacy by 1.5-2.5 times.

D. Flow Analysis Algorithm

We have so far used a Boolean variable $f(p)$ to denote whether a VoIP client p can be a potential receiver for a VoIP call from src. In this section, we use the flow measurements to construct a probability distribution over the set of possible receivers. Let G^κ be a sub-graph of G obtained using the top-κ shortest path tracing algorithm with caller src. Let $nf(p \rightarrow q)$ denote the number of flows on the edge $p \rightarrow q$. Let $in(p)$ denote the total number of flows into node p. Note that both $nf(p \rightarrow q)$ and $in(p)$ are observable by an external adversary. Assuming a node p in the VoIP network performs perfect mixing, the probability that some incoming flow is forwarded on the edge $p \rightarrow q$ as observed by an external adversary is $\frac{nf(p \rightarrow q)}{in(p)}$. Let $f(p)$ denote the probability that a VoIP flow originating at src flows through node p. The function f is recursively defined on the directed edges in G^κ=$\langle V^\kappa, E^\kappa \rangle$ as follows:

$$f(q) = \sum_{p \rightarrow q \in E^\kappa} f(p) * \frac{nf(p \rightarrow q)}{in(p)} \qquad (1)$$

with the base case $f(src)$ = 1 and $in(src)$ = 1. Now, every VoIP client p ($p \neq src$) is a possible destination for the VoIP flow originating from src if $f(p) > 0$. We use the top-m probability metric, namely, the probability that the receiver dst appears in the top-m entries when $f(p)$ is sorted in descending order. Top-m probability besides being an intuitive metric directly relates to the information theoretic notion of self-information (entropy) $I = -\log p$. Indeed higher entropy represents higher randomness and thus translates into better privacy. Such probability and entropy based metrics are often used for quantifying privacy in anonymous networks [19].

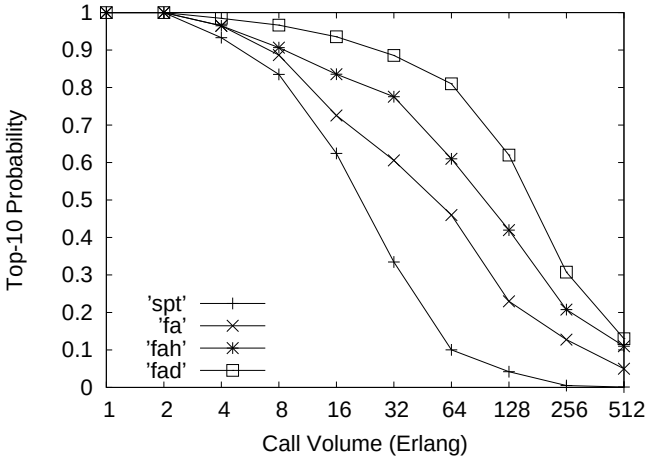

Fig. 13. Top-10 Probability (κ = 2): Comparison between shortest path tracing, flow analysis algorithm with no prior and Distance prior

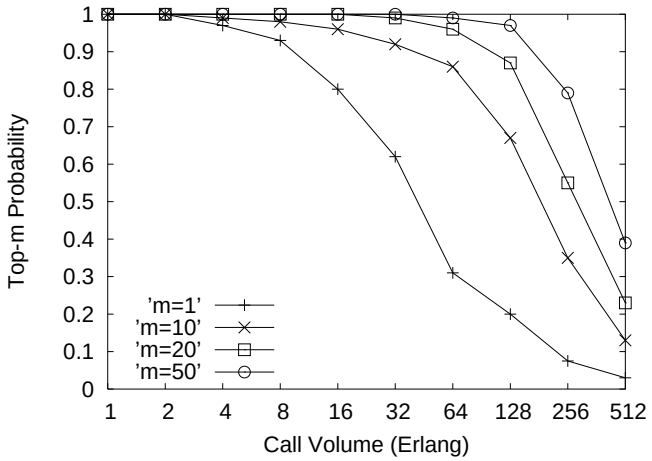

Fig. 14. Top-m Probability with κ = 2 and Distance prior

κ	# Iterations	Time (ms)
1	-	0.03
2	7	31
3	11	49.7
4	19	84.2

Fig. 15. Computation Cost

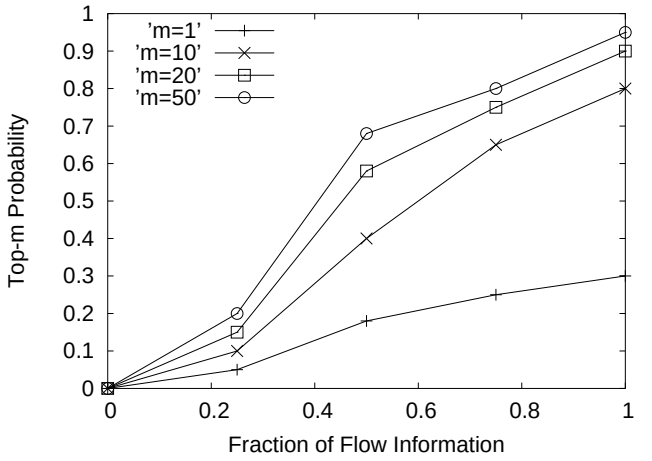

Fig. 16. Incomplete Flow Information

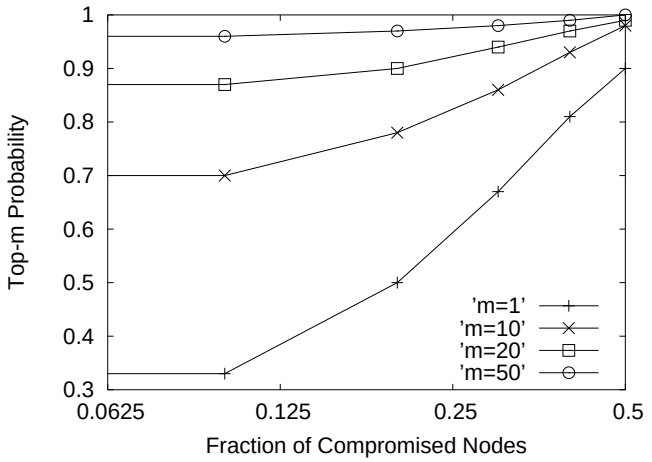

Fig. 17. Compromised Proxies

Computing the probabilities $f(p)$ for G^1 (top-1 shortest paths) is very efficient. Observe that since G^1 is a directed acyclic graph, it can be sorted topologically. Let $p_1 = src$, p_2, $\cdots$ p_N be a topological ordering of the nodes in G^1 such that $f(p_i)$ depends on $f(p_j)$ only if $j < i$. Hence, one can efficiently evaluate the probabilities by following the topological order, namely, compute $f(p_2)$, $f(p_3)$, $\cdots$, $f(p_N)$ in that order starting with $f(p_1) = 1$.

However, G^κ (for $\kappa \geq 2$) may contain cycles and thus cannot be topologically sorted. In this case, we represent the set of equations in 1 as $\pi = \pi M$, where π is a $1 \times N$ row vector and M is a $N \times N$ matrix, where $\pi_i = f(p_i)$ and $M_{ij} = \frac{nf(p_i \to p_j)}{in(p_i)}$ if there exists a directed edge $p_i \to p_j$ in G^κ; and $M_{ij} = 0$ otherwise. Hence, the solution π is the stationary distribution of a Markov chain whose transition probability matrix is given by M. We compute π using a *fixed point computation* approach as follows: we recursively compute $\pi^{t+1} = \pi^t M$ starting with $\pi_i^0 = 1$ if $p_i = src$; $\pi_i^0 = 0$ otherwise. Assuming M is irreducible, π converges to a steady state solution π^* in $O(N \log N)$ iterations. If the matrix M were not irreducible (this happens if the underlying directed graph G^κ is not strongly connected[5]), then we approximate π^* as $\frac{\sum_{t=0}^{\tau * N \log N} \pi^t}{\tau * N \log N}$ for some constant $\tau \geq 1$ (typically, we set $\tau = 1$).

E. Distance Prior and Hop Count Prior

In this section, we enhance the efficacy of the flow analysis algorithm using hop count and distance *prior*. We use g_{hop} and g_{lat} to denote hop count and distance (in terms of latency) between src and dst. For instance, one can use g_{hop} and g_{lat} to encode the fact that most calls are between nodes in the same autonomous system. Using the hop count prior, the probability that a node p forwards an incoming flow on the edge $p \to q$ is $\frac{nf(p \to q)}{in(p)} * Pr(hop \geq hc(src, p) + 1 \mid hop \geq hc(src, p))$, where $hc(src, p)$ denotes the number of hops along the shortest path between src and p on graph G^κ. $Pr(hop \geq$

[5]A directed graph is said to be strongly connected if there exists a directed path from every vertex to every other vertex in the graph

$hc(src,\ p)\ +\ 1\ \mid\ hop\ \geq\ hc(src,\ p))\ =\ \frac{Pr(hop \geq hc(src,p)+1)}{Pr(hop \geq hc(src,p))}$ denotes the probability that the receiver dst could be one more hop away given that dst is at least $hc(src, p)$ hops away from src.

A similar analysis applies to distance prior as well. We use $dist(src, p)$ to denote the latency of the shortest path between src and p on graph G^κ, and $w(p, q)$ denotes the one-way latency between nodes p and q. As with the flow analysis algorithm, the function f is defined on the directed edges in graph $G^\kappa = \langle V^\kappa,\ E^\kappa \rangle$ as follows:

$$f(q) = \sum_{p \rightarrow q \in E^\kappa} f(p) * \frac{nf(p \rightarrow q)}{in(p)} * \frac{Pr(hop \geq hc(src,p)+1)}{Pr(hop \geq hc(src,p))}$$

$$f(q) = \sum_{p \rightarrow q \in E^\kappa} f(p) * \frac{nf(p \rightarrow q)}{in(p)} * \frac{Pr(lat \geq dist(src,p)+w(p,q))}{Pr(lat \geq dist(src,p))}$$

Figure 13 shows the Top-10 probability of an attack versus call volume using κ = 2 for different attack algorithms `spt`: statistical shortest path tracing, `fa`: flow analysis (with no priors), `fah`: flow analysis with hop count prior and `fad`: flow analysis with distance prior. We observe that the flow analysis algorithm with distance prior offers the best results. This is primarily because the route setup protocol always constructs voice paths that have minimum one-way latency. The distance prior directly reflects on the latency based shortest path nature of the route setup protocol and thus performs best.

Figure 14 shows the top-m probability, namely, the probability that the true receiver dst appears in the top-m entries when $f(p)$ is sorted in descending order using the flow analysis algorithm with distance prior and κ = 2. We also experimented with hop count prior; however, distance prior directly reflects on the latency based shortest path nature of the route setup protocol and thus performs best. With a call volume of 64 Erlangs, there is 86% chance that the true receiver dst appears in the top-10 entries. Under very high call volume (512 Erlangs) the top-10 probability drops to 0.17. However, we note from our enterprise data set (see Fig 4) that the call volume is smaller than 64 Erlangs for about 75% of the day.

Figure 15 shows the computation cost[6] incurred in computing the probabilities $f(p)$ for all potential receivers p. In practice, we observed that the number of iterations required for $f(p)$ to converge to its stationary value is much smaller that the theoretical bound $O(N \log N)$. We attribute this to the sparse nature of matrix M. The overall running time is of the order of few tens of milliseconds making the attack feasible in real-time.

F. Incomplete Flow Information

So far, we have assumed that the adversary can monitor the flow rate on all the links in the VoIP network. In the absence of flow information on a link $p \rightarrow q$, we use an unbiased random walk estimator to compute $f(p)$'s contribution to $f(q)$ as $f(p) * \frac{1}{deg(p)}$, where $deg(p)$ denotes the number of neighbors of p on the VoIP network. As with the flow analysis algorithm, the function f is defined on the directed edges in graph $G^\kappa = \langle V^\kappa,\ E^\kappa \rangle$ and the set of unmonitored links $U \subseteq E^\kappa$ as follows:

$$f(q) = \sum_{p \rightarrow q \in U} f(p) * \frac{1}{deg(p)} + \sum_{p \rightarrow q \in E^\kappa \setminus U} f(p) * \frac{nf(p \rightarrow q)}{in(p)}$$

Figure 16 shows the efficacy of the attack assuming that only a fraction of the flow information is monitored by the adversary. It follows from the figure that with only 20% flow information, flow analysis attacks are ineffective; however, with 30-60% flow information the top-m probability increases substantially. This suggests that the flow analysis attack is robust against some inaccuracies in topology and flow information.

G. Compromised Proxies

In addition, to passive observation based attacks, the adversary could actively compromise some of the nodes in the VoIP proxy. We assume an honest-but-curious model for the compromised nodes. A compromised node p reveals its mixing information (namely, $nf(r \rightarrow p \rightarrow q)$) to an adversary, where $nf(r \rightarrow p \rightarrow q)$ denotes the number of voice flows that were routed from r to q by the malicious node p. With slight abuse of notation, we use $f(p \rightarrow q)$ to denote the probability that a VoIP call from src traverses the edge $p \rightarrow q$. Hence, the new flow analysis equations are as follows:

$$\begin{aligned} f(p \rightarrow q) &= \begin{cases} f(p) * \frac{nf(p \rightarrow q)}{in(p)} & \text{honest } p \\ \sum_{r \rightarrow p} f(r \rightarrow p) * \frac{nf(r \rightarrow p \rightarrow q)}{nf(r \rightarrow p)} & \text{malicious } p \end{cases} \\ f(q) &= \sum_{p \rightarrow q} f(p \rightarrow q) \end{aligned} \quad (2)$$

Figure 17 shows the effectiveness of the attack as we vary the fraction of nodes that is compromised by the adversary. We use a call volume of 128 Erlangs and κ = 2 in this experiment. Evidently, compromised proxy nodes significantly enhance attack efficacy; for instance, when 20% of the nodes are compromised the top-1 probability improves from 0.23 to 0.50.

IV. VoIP Privacy using k-Anonymity

In this section, we develop a k-anonymity approach to protect the identity of a receiver from flow analysis attacks. We define k-anonymity for identical voice flows as follows:

k-anonymity: A voice flow from src to dst is said to be k-anonymous if the size of a candidate receiver set identified by an adversary using the naive tracking algorithm is no smaller than k[7].

As pointed out earlier, k-anonymity can be realized by mixing a voice flow from src to dst with at least $k-1$ other flows (each of which has a different source and destination nodes). The primary supposition that any two voice flows are indistinguishable under traffic analysis (see Fig 2) follows from the properties of RTP (real-time protocol) and the constant packet rate property of silence suppression free audio codecs. One

[6] As measured on a 900 MHz Pentium III processor running RedHat Linux 7.2

[7] k-anonymity is defined with respect to naive tracing, since AARSP does not use shortest path routing

way to achieve k-anonymity is to mix a flow from src to dst with $k-1$ dummy voice flows; however, this approach can increase aggregation bandwidth consumption by k-fold. In this section, we propose an anonymity aware route set up protocol (AARSP). AARSP reroutes and mixes *existing* voice flows (without adding dummy traffic) with the goal of: (i) meeting k-anonymity, and (ii) satisfying latency based QoS guarantee.

Figures 18 illustrates a simple scenario with two VoIP flows between clients (s_1, d_1) and (s_2, d_2). In the figures, each edge on the VoIP network is marked with the number of flows it carries. Figure 19 shows how one can reroute VoIP traffic in order to achieve 2-anonymity. Evidently, the rerouted flows may not satisfy the shortest route property and thus may violate latency based QoS guarantees. However, AARSP exploits the slack between the shortest path latency and the tolerable latency (250ms) to accomplish its goals.

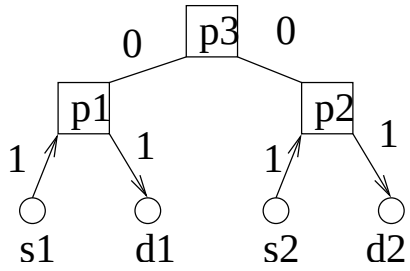

Fig. 18. 1-anonymity

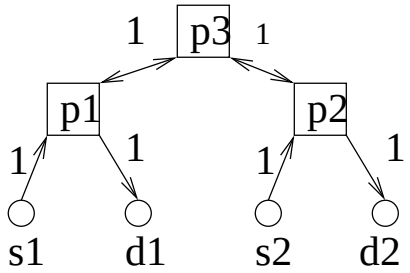

Fig. 19. 2-anonymity

A. AARSP: Anonymity-Aware RSP

In this section, we summarize our ideas behind AARSP. AARSP accepts an anonymity parameter k as an input for the route setup protocol, on a per-client per-call basis. AARSP modifies the basic route set up protocol (RSP) such that it simultaneously satisfies three conditions: (i) we have at least one node $p \in route(src, dst)$ such that $in(p) \geq k$ (k-anonymity), (ii) the end-to-end one-way latency on the route from src to dst is smaller than $maxLat$ (typically set to 250ms), and (iii) the total call volume on every node $p \in route(src, dst)$ is smaller than its capacity $maxFlow(p)$.

Similar to the naive route set up protocol discussed in Section II, AARSP uses an expanding search approach to construct a k-anonymous path from a caller src to a recipient dst. The key idea is that the expanding search not only tracks the distance from src to a node p in the network ($dist[p]$), but also the anonymity of a route from src to p ($anon[p]$). Recall that the naive route set up protocol drops a search request if it has seen the search identifier $searchId$ in the recent past. On the other hand, in the AARSP protocol, let us suppose that a node p receives a search request identified by $searchId$ at two time instants t_1 and t_2 (wlog, $t_1 < t_2$) such that the anonymity of the route traversed by the search requests is k_1 and k_2 respectively. Node p drops the search request received at time t_2 if: $k_1 \geq k$ or $k_1 \geq k_2$.

AARSP may intentionally introduce loops in the path to improve the anonymity of voice flows. In doing so, AARSP ensures the protocol converges to a valid path by limiting number of times any loop is traversed by a path to at most one. Unlike the shortest path route setup protocol, the routing entry at a node p in the AARSP protocol is a three tuple $\langle sipurl, prev, next \rangle$, which indicates that when a node p receives voice packets from node $prev$ destined to $sipurl$, then node p must forward the packet to node $next$. The three tuple routing table allows us to handle loops; for example, from Figure 19 node p_3 may have the following two routing entries: $\langle d_1, p_1, p_2 \rangle$ and $\langle d_1, p_2, p_1 \rangle$. In the AARSP protocol, let us suppose that a node p receives a search request identified by $searchId$ at two time instants t_1 and t_2 (wlog, $t_1 < t_2$) such that the search request was forwarded to node p by nodes $prev_1$ and $prev_2$ respectively. Node p drops the search request received at time t_2 if: $prev_1 = prev_2$.

AARSP sets up the shortest possible route from src to dst that meets k-anonymity (if there exists such a route). If no such route exists, AARSP setup identifies a route with highest possible anonymity $k' < k$; in addition, AARSP sets up the shortest route that achieves k'-anonymous route from src to dst. Indeed, if one sets $k = \infty$, then AARSP identifies the most anonymous route whose one-way latency is smaller than $maxLat$ = 250ms. For detailed analysis on the properties of the AARSP protocol we refer the readers to a detailed technical report [27].

Figure 20 compares the average one-way path latency for both AARSP and RSP for varying call volumes. At low call volumes AARSP may construct significantly longer routes than RSP. At lower call volumes, AARSP may significantly deviate from the shortest path to achieve the desired level of

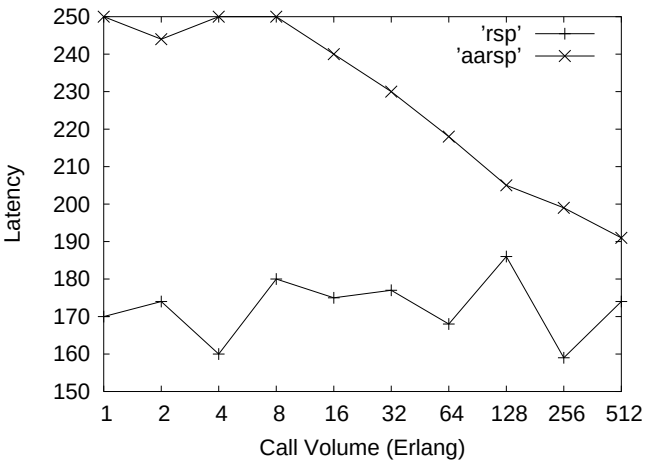

Fig. 20. AARSP Vs RSP: Path Latency

anonymity; higher call volumes facilitate natural mixing of voice flows and thus require relatively smaller deviation from the shortest path. Nonetheless, AARSP ensures that the latency is below 250ms and thus preserves the quality of voice conversations.

Figure 21 shows the anonymity level of a VoIP call as a function of its hold time given that the initial route was established using k = 20. Anonymity level of a VoIP flow F at time t denotes the number of flows mixed with F at time t along F's route. We tracked the worst case (lowest) and the best case (highest) anonymity levels of a VoIP call. We observe that at lower call volumes (64 Erlangs) calls with hold time larger than 400 seconds may experience time durations at which their anonymity level falls below 80% of the initial value. Fortunately, 95% of voice calls are shorter than 400s (see Fig 5). Long lasting calls may set up a lower watermark level $k' \leq k$ such that the AARSP route maintenance protocol automatically terminates the call when its anonymity level falls below k'.

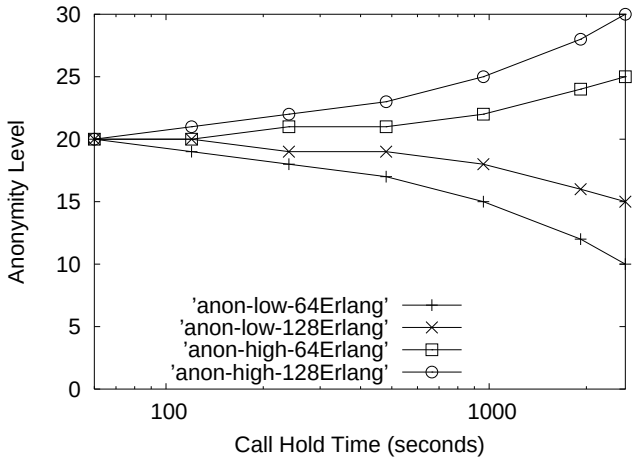

Fig. 21. Route Maintenance

B. Flow Analysis Attacks on AARSP

A flow analysis attack on AARSP operates efficiently as follows. We assume that the adversary knows the parameter k used by a caller src. First, the adversary identifies all nodes p such that $in(p) \geq k$; if there exists no such p, then the attacker picks p that maximizes $in(p)$. Let G^{κ}_{src} be a sub-graph of G obtained using the top-κ shortest path tracing algorithm with caller src. We use G^{κ}_{src} and the flow measurements to compute the probability that the voice flow is routed from src to p ($f_{src}(p)$) for all p such that $in(p) \geq k$ starting with $f_{src}(src)$ = 1. The algorithm used for computing $f_{src}(p)$ is described in Section III-D.

Second, for every such node p with $in(p) \geq k$, we construct G^{κ}_{p} as a sub-graph of G obtained using the top-κ shortest path tracing algorithm with p as the caller. We compute the probability of a voice flow being routed from src to r via p ($f_{src,p}(r)$) for every candidate receiver r starting with $f_{src,p}(p) = f_{src}(p)$, where $f_{src}(p)$ is obtained from the first step. Third, we compute the probability of r being the receiver as $f_{recv}(r) = f_{src,pivot}(r)$, where $pivot = argmin_p \{dist(src, p) + dist(p, r)\}$. Similar to other flow analysis attacks, one could sort the receivers in descending order of $f(r)$ and use a top-m probability metric to study the efficacy of the attack.

Figures 22 and 23 compare the effectiveness of AARSP with the shortest path route setup protocol (RSP) in mitigating flow analysis attacks. We set the parameter $k = \infty$ so that AARSP identifies the *most anonymous* route from caller src to receiver dst that satisfies the latency constraint. We use the flow analysis described in this section for AARSP and a flow analysis attack with distance prior for RSP. Figure 22 shows the top-10 probability using both AARSP and RSP for varying call volumes. Observe that at low and moderate call volumes AARSP offers significantly improved protection against flow analysis attacks. Figure 23 shows the top-m probability for both AARSP and RSP for varying m and a call volume of 128 Erlangs. We also observe that AARSP consistently out performs RSP for all values of m.

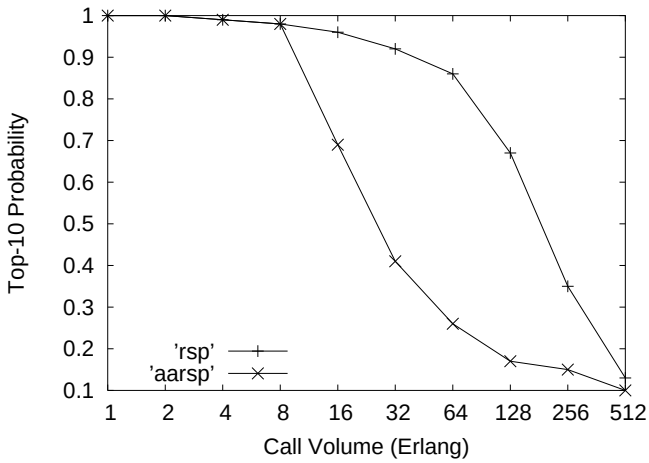

Fig. 22. AARSP Vs RSP: Top-10 Probability

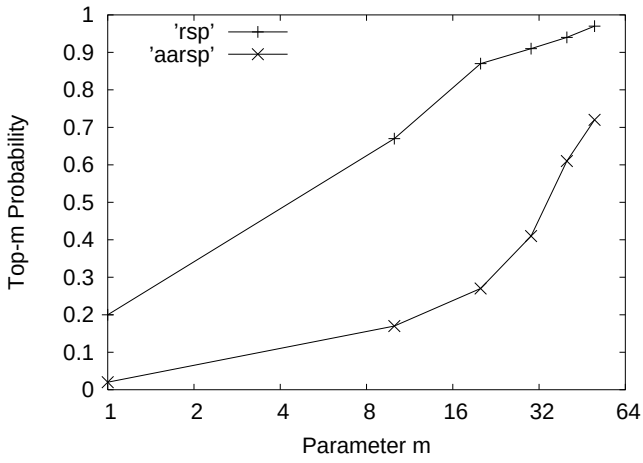

Fig. 23. AARSP Vs RSP: 128 Erlangs

C. Tolerating Compromised Proxies

In this section, we study the effect of compromised proxies on AARSP. Similar to Section III, we assume that a compromised node will execute AARSP honestly. However, the malicious nodes may record observed information during the voice session and collude with the external observer. In particular, a malicious node may reveal the mapping between its in-flows and out-flows thereby decreasing the anonymity of a VoIP flow.

We describe a simple extension to AARSP to tolerate malicious nodes and yet preserve k-anonymity. We allow the caller src to specify a personalized security parameter c for every VoIP call indicating that the route from src to dst should tolerate up to c compromised nodes while preserving k-anonymity. The key idea is to construct a route from src to dst with at least $c+1$ nodes p_1, p_2, $\cdots$, p_{c+1} such that $in(p_i) \geq k$ for all $1 \leq i \leq c+1$. One can introduce this constraint by recording top-c anonymity levels in the $anon$ field of ASRSP. Now, one could replace the constraint $anon \geq k$ with $anon \geq (k, c)$.

While this approach offers significantly higher attack resilience, one can extend the flow analysis attack on a k-anonymous AARSP to a (k, c)-anonymous AARSP. We assume that the adversary knows the parameters k and c used by a caller src. The adversary identifies a set of nodes S such that for all nodes $p \in S$, $in(p) \geq k$. For every permutation of c nodes from the set S, the adversary computes the probability of a voice flow being routed from src to r via p_1, p_2, $\cdots$, p_c ($f_{src,p_1,p_2,\cdots,p_c}(r)$) for a candidate receiver r. We recursively compute $f_{src,p_1,p_2,\cdots,p_i}(r)$ using flow analysis algorithms on $G^{\kappa}_{p_i}$ and starting probability $f_{src,p_1,\cdots,p_i}(p_i) = f_{src,p_1,p_2,\cdots,p_{i-1}}(p_i)$. The recursion terminates at the base case $f_{src}(src) = 1$. Finally, we compute the probability of r being the receiver as $f_{recv}(r) = f_{src,pivot}(r)$, where $pivot = argmin_{\{p_1,p_2,\cdots,p_c\} \subseteq S}\ \{latdist(src, p_1) + latdist(p_1, p_2) + \cdots + latdist(p_{c-1}, p_c) + latdist(p_c, r)\}$. This attack reduces a simple flow analysis attack on AARSP when $c = 1$; however, the $pivot$ size and consequently the attack complexity grows exponentially in c.

Figure 24 shows the effectiveness of the attack as we vary the fraction of nodes that are compromised by the adversary and the parameter c. We use a call volume of 128 Erlangs, $k = 10$ and $\kappa = 2$ in this experiment. We observe that as c increases the effectiveness of the attack decreases significantly; for instance, when 10% of the nodes are compromised, using $c = 3$ reduces the top-10 probability to 0.06 when compared to $c = 1$ which results in a top-10 probability of 0.44. Nonetheless, when a large number of nodes are malicious AARSP becomes vulnerable to flow analysis attacks. Under a realistic setting, when only a small number of nodes ($< 20\%$) are likely to be malicious, AARSP offers very good protection against flow analysis attacks.

Figure 25 shows the computation cost incurred to an adversary as k increases. As k increases, the number of pivot nodes p, that is, p such that $in(p) \geq k$ decreases; and thus, the computation cost decreases. Figure 26 shows the computation cost for an attack as c increases with k=∞. We observe that the computation cost for an attack grows rapidly with c, making it harder for an adversary to launch flow analysis attacks in real-time.

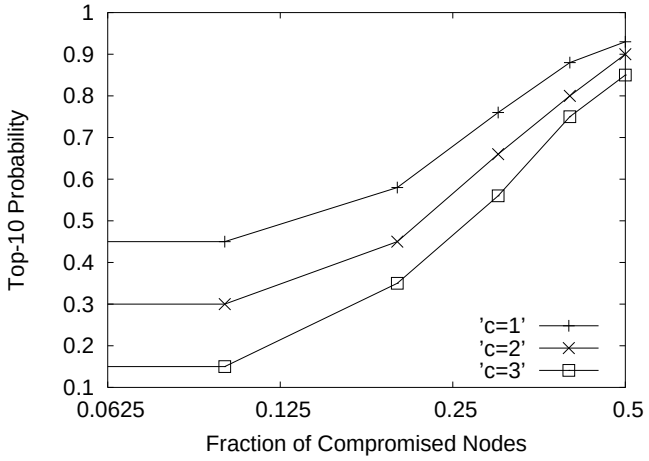

Fig. 24. Compromised Proxies

k	Time(s)
2	9.0
10	5.9
20	4.0
50	2.9

Fig. 25. Attack Cost Vs Anonymity Level (k)

V. Performance Evaluation

A. Implementation Sketch

In this section, we briefly describe an implementation of our algorithms using Phex [4]: an open source Java based implementation of shortest route set up protocol (RSP). VoIP protocols operate on top of the peer-to-peer infrastructure. We have implemented our algorithms as pluggable modules that can be weaved into the Phex client code using AspectJ [12]. Our implementation is completely transparent to the VoIP protocol that operates on top of the peer-to-peer infrastructure. Also, our implementation does not require any changes to topology construction and maintenance algorithms (as nodes join, leave, fail or recover) and the underlying TCP/IP or UDP based communication libraries.

Below we sketch our implementation of AARSP. The Phex broadcast search protocol has four operations: `initSearch`, `processSearch`, `processResult` and `finSearch`. These four operations are implemented as event handlers in Phex. When a Phex client receives a messages, it determines the type of the message (search request, search result, etc) and

c	Time(s)
1	2.9
2	4.4
3	11.6
4	43.5

Fig. 26. Attack Cost Vs Tolerance to Malicious Nodes (c)

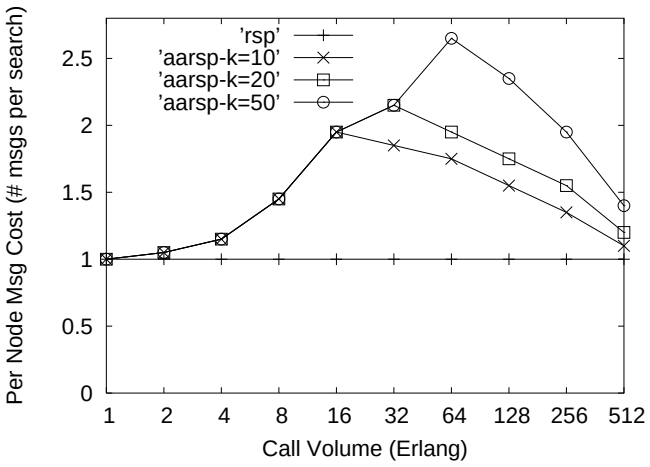

Fig. 27. Messaging Cost

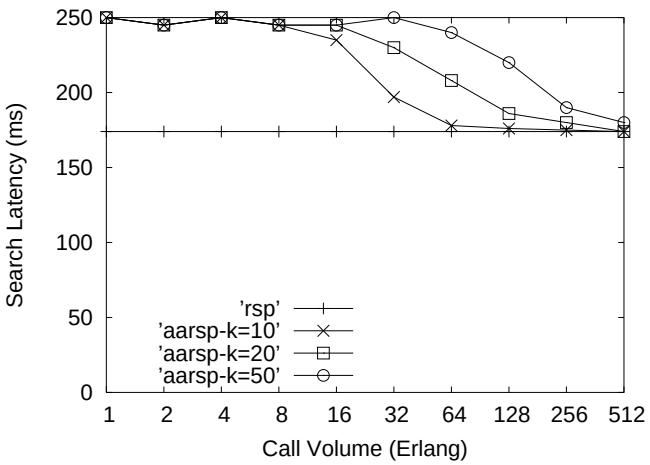

Fig. 28. Search Latency

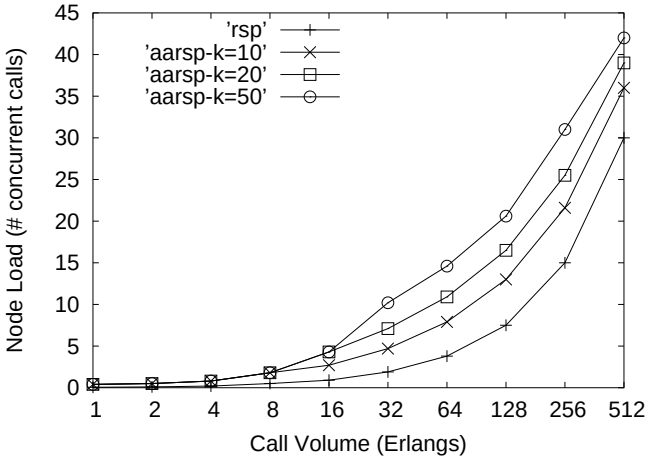

Fig. 29. Node Load

triggers the appropriate event handler. AARSP substitutes all four event handlers (see Section IV). In addition, we also modify the search request payload and the search result payload to include the parameters k and c. In the rest of this section, all experimental results are reported using our implementation deployed on 64 nodes in Planet Lab[8].

B. Performance and Scalability

In this section, we compare the messaging cost and the search latency of AARSP and RSP. Figure 27 shows the messaging cost per node as we vary the call volume and the anonymity parameter k (using c = 1). These experiments show that AARSP incurs about 1-3 times the messaging cost of RSP. However, the search request and the results are typically of the order of 300 Bytes. Hence, the communication cost at a node for handling 10 messages (3 KB) is equivalent to a voice session of one second (24 Kbps). Even though AARSP incurs higher communication cost than RSP, its effect on the overall bandwidth consumption is negligible.

Figure 28 shows the latency of a search operation as we vary the call volume and the anonymity parameter k (using c = 1). This experiment shows that AARSPs incurs about 30-40% higher search latency than RSP. One should note that the search latency only affects the initial connection set up time. Once the route is established AARSP ensures good quality voice conversations by limiting the path latency to 250ms. Figures 27 and 28 also show that the relative overhead of AARSP over RSP decreases with call volume. The key intuition here is that higher call volumes facilitate natural mixing of voice flows, thereby decreasing the overall messaging and search cost.

Figure 29 shows the average number of concurrent VoIP calls handled by a node in the VoIP network (using c = 1). AARSP incurs a higher load primarily because the traffic is not routed through the optimal route. Hence, a voice route in AARSP may include more network hops than RSP. This increases the average number of proxies that route one voice call, and thus increases the average load on a proxy. The percentile increase in average node load for higher call volumes is small. Hence, when the call volume is high (VoIP network is heavily loaded), AARSP imposes small overheads (20-40%) compared with RSP. On the other hand, when the call volume is low (VoIP network is lightly loaded), AARSP incurs 2-3 times the average node load when compared to RSP. However, this 2-3x increase in node load is incurred when the VoIP network is itself lightly loaded; hence, AARSP does not harm the performance and scalability of the VoIP network.

VI. Related Work

Privacy has long been a hot button issue for both the VoIP clients and the law enforcement bodies. On one hand, users want their phone conversations to be anonymous; anonymity offers them *possible deniability* thereby shielding them from law enforcement bodies. On the other hand FCC (Federal

[8]http://www.planet-lab.org/

Communications Commission) [13] considers capability of tracking VoIP calls "of paramount importance to the law enforcement and the national security interests of the United States". Similar to other VoIP privacy papers [30], we leave aside the controversy between anonymity and security. Instead we focus on technical feasibility of privacy attacks and defenses on VoIP networks.

Mix [9] is a routing element that attempts to hide correspondences between its input and output messages. A large number of low latency anonymizing networks have been built using the concept of a mix network [9][25]. Onion routing [16] and its second generation Tor [10] aim at providing anonymous transport of TCP flows over the Internet. ISDN mixes [20] proposes solutions to anonymize phone calls over traditional PSTN (Public Switched Telephone Networks). In this paper we have focused on VoIP networks given its recent wide spread adoption[9].

It is widely acknowledged that low latency anonymizing networks [10][16][7] are vulnerable to timing analysis attacks [28][25], especially from well placed malicious attackers [33]. Several papers have addressed the problem of tracing encrypted traffic using timing analysis [32][34][36][31][8][11][30]. All these papers use inter-packet timing characteristics for tracing traffic. Complementary to all these approaches, we have introduced flow analysis attacks that target the shortest path property of voice routes and presented techniques to provide customizable anonymity guarantees in a VoIP network. Unlike the timing analysis attacks, our approach does not rely upon inter-packet times to detect caller-receiver pairs; instead we analyze the volume of flow in the VoIP network and deduce possible caller-receiver pairs using the flow information and the underlying VoIP network topology.

Tarzan [15] presents an anonymizing network layer using a gossip-based peer-to-peer protocol. We note that flow analysis attacks target the shortest path property and not the protocol used for constructing the route itself; hence, a gossip based shortest path setup protocol is equally vulnerable to flow analysis attacks.

Traditionally, multicast and broadcast protocols have been used to protect receiver anonymity [21][24]. However, in a multicast based approach achieving k-anonymity may increase the network traffic by k-fold. In contrast our paper attempts to reroute and mix existing voice flows and thus incurs significantly smaller overhead on the VoIP network.

VII. Conclusion

In this paper we have addressed the problem of providing privacy guarantees in peer-to-peer VoIP networks. First, we have developed flow analysis attacks that allow an adversary (external observer) to identify a small and accurate set of candidate receivers even when all the nodes in the network are honest. We have used network flow analysis and statistical inference to study the efficacy of such an attack. Second, we have developed mixing based techniques to provide a guaranteed level of anonymity for VoIP clients. We have developed an anonymity aware route setup protocol (AARSP) that allows clients to specify personalized privacy requirements for their voice calls (on a per-client per-call basis) using a quantifiable k-anonymity metric. We have implemented our proposal on the Phex client and presented detailed experimental evaluation that demonstrates the performance and scalability of our protocol, while meeting customizable privacy guarantees.

[9]According to TeleGeography Research [6], worldwide VoIP's share of voice traffic has grown from 12.8% in 2003 to about 44% in 2007

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] A. Back, I. Goldberg, and A. Shostack. Freedom 2.1 security issues and analysis. Zero Knowledge Systems, Inc. white paper, 2001.
[8] A. Blum, D. Song, and S. Venkataraman. Detectiong of interactive stepping stones: Algorithms and confidence bounds. In *7th RAID*, 2004.
[9] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of ACM, 24(2): 84-88*, 1981.
[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second generation onion router. In *13th USENIX Security Symposium*, 2000.
[11] 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.
[12] Eclipse. Aspectj compiler. http://eclipse.org/aspectj.
[13] FBI. Letter to FCC. http://www.askcalea.com/docs/20040128.jper.letter.pdf.
[14] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. In *IEEE Journal on Special Areas in Communication*, 2002.
[15] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM CCS*, 2002.
[16] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. In *Communications of ACM, Vol 42(2)*, 1999.
[17] 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.
[18] J. Y. Hui. Switching and traffic theory for integrated broadband networks. Academic Press ISBN: 079239061X, 1990.
[19] G. Perng, M. K. Reiter, and C. Wang. M2: Multicasting mixes for efficient and anonymous communication. In *IEEE ICDCS*, 2006.
[20] 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.
[21] A. Pfitzmann and M. Waidner. Networks without user observability. In *Computers and Security, 2(6): 158-166*.
[22] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *12th IEEE INFOCOM*, 2001.
[23] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networks (MMCN)*, 2002.
[24] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *ACM CCS*, 2000.
[25] V. Shmatikov and M. H. Wang. Timing analysis in low latency mix networks: Attacks and defenses. In *11th ESORICS*, 2006.
[26] G. I. Sound. VoIP: Better than PSTN? http://www.globalipsound.com/demo/tutorial.php.
[27] M. Srivatsa, A. Iyengar, and L. Liu. Privacy in voip networks: A k-anonymity approach. Technical report, IBM Research RC24625, 2008.
[28] 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.
[29] C. J. Van-Rijsbergen. Information retrieval. In *Second Edition, Butterworths, London*, 1979.
[30] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the internet. In *12th ACM CCS*, 2005.
[31] X. Wang and D. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *10th ACM CCS*, 2003.

[32] X. Wang, D. Reeves, and S. Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *7th ESORICS*, 2002.
[33] 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.
[34] K. yoda and H. Etoh. Finding a connection chain for tracing intruders. In *6th ESORICS*, 2000.
[35] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork? In *IEEE Infocom*, 1996.
[36] Y. Zhang and V. Paxon. Detecting stepping stones. In *9th USENIX Security Symposium*, 2000.

Mudhakar Srivatsa is a research scientist at IBM T. J. Watson Research Center where he conducts research at the intersection of networking and security. He received a B.Tech degree in Computer Science and Engineering from IIT, Madras, and a PhD degree in Computer Science from Georgia Tech. His research interests primarily include security and reliability of large scale networks, secure information flow, and risk management.

Arun Iyengar received the Ph.D. degree in computer science from MIT. He does research and development into Web performance, distributed computing, and high availability at IBM's T.J. Watson Research Center. Arun is Co-Editor-in-Chief of the ACM Transactions on the Web, Founding Chair of IFIP Working Group 6.4 on Internet Applications Engineering, and an IBM Master Inventor.

Ling Liu is an associate professor at the College of Computing at the Georgia Institute of Technology. There, she directs the research programs in the Distributed Data Intensive Systems Lab (DiSL), examining research issues and technical challenges in building large scale distributed computing systems that can grow without limits. Her research group has produced a number of software systems that are either open sources or directly accessible online, among which the most popular are WebCQ and XWRAPElite. She is on the editorial board of several international journals, such as the IEEE Transactions on Knowledge and Data Engineering, the International Journal of Very large Database Systems (VLDBJ), and the International Journal of Web Services Research.

Hongbo Jiang is an associate professor at the Department of Electronics and Information Engineering at the Huazhong University of Science and Technology. Hongbo leads Netwoked and Communication Systems Research group, wherein he conducts research on computer networking, especially focused on algorithms and architectures for high performance wireless networks.