Privacy in VoIP Networks: A k-Anonymity Approach

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

***Abstract*— Peer-to-Peer VoIP (voice over IP) networks, exemplified by Skype [4], 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 present detailed experimental evaluation that demonstrates these attacks quantifying performance and scalability degradation.**

I. Introduction

The concept of a mix [8] 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 [13], Tor [9], Tarzan [12], or Freedom [6]. 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[1].

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). These networks allow 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.

[1]Recent case study [22] indicates that latencies up to 250ms are unperceivable to human users; while latencies over 400ms significantly deteriorate the quality of voice conversations

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

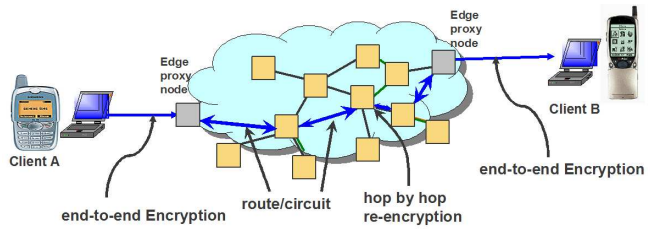

Fig. 1. Anonymizing VoIP Network

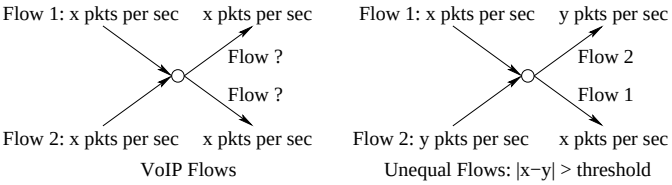

Fig. 2. Mixing Statistically Identical VoIP Flows

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.723A 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 II. We present related work in Section III and finally conclude in Section IV.

II. 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

[3]G.723A 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

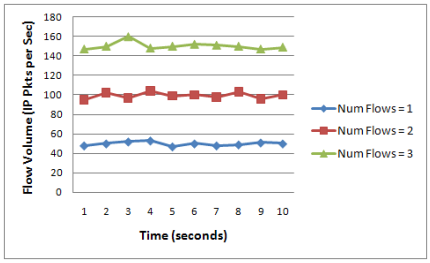

Fig. 3. Inferring Number of Flows from Flow Volume

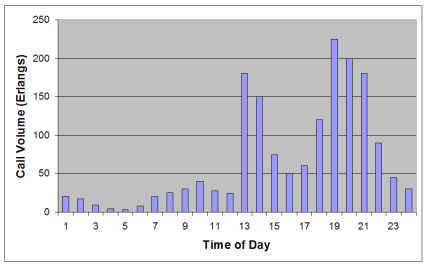

Fig. 4. Call Volume Data

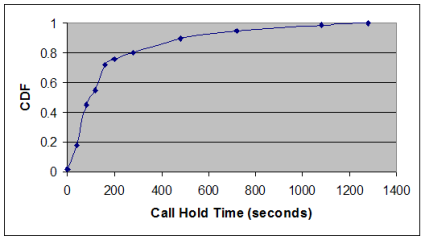

Fig. 5. Call Hold Time Data

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 [23][9] and the flow rates on *all* links in the VoIP network [16][6][4] (see Fig 3). We experimentally show that the attack can be very effective even when only 45-65% 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 [30][1] and our experiments were performed on NS-2 [2][3]. GT-ITM models network geography (stub domains and autonomous systems) and the small world phenomenon (power law graph with parameter γ=2.1) [11][19]. 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 [14]. The average route (shortest path) latency between any two nodes in the network is 170ms, while the worst case route latency is 225ms.

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 [15]: 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; hence, we assume that for a given VoIP call, the (src, dst) pair is chosen randomly from the VoIP network. As we note in Section II-E, any prior information (such as, 80% of the calls are between nodes in the same autonomous system) can be used by the adversary to further enhance the efficacy of flow analysis attacks.

[4]The network service provider can obtain VoIP topology and flow information using traffic analysis

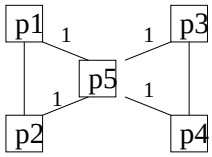

Fig. 6. Tracing Voice Calls

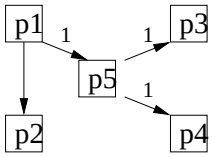

Fig. 7. Shortest Path Tracing

TRACE(Graph G=$\langle V, E\rangle$, Caller src)
(1) **for each** vertex $v \in V$
(2) $f[v]$ = 0; $label[v]$ = `false`
(3) **end for**
(4) $f[src]$ = 1; $label[src]$ = `true`
(5) **while** pick a labeled vertex v
(6) $label[v]$ = `false`
(7) **for each** node u such that $(u, v) \in E$
(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

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 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.

SHORTEST PATH TRACING(Graph G=$\langle V,\ E\rangle$, Caller src)

(1) **for each** vertex $v \in V$
(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 $(u, v) \in E$
(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) $prev[u] = prev[u] \cup \{v\}$
(14) **end if**
(15) **end for**
(16) **end while**
(17) $G^1 = \langle V^1, E^1\rangle$: $V^1 = V$, $E^1 = (u \rightarrow v)\ \forall u \in prev[v], \forall v \in V$

Fig. 9. Shortest Path Tracing Algorithm

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 Erlang. 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 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 [24].

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

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.

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; and the 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 become 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

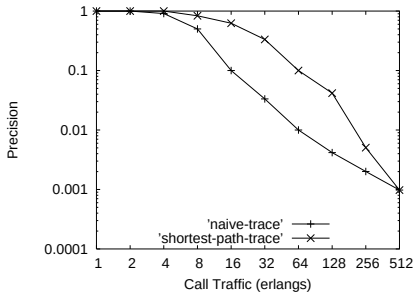

Fig. 10. Shortest Path Tracing Vs Naive Tracing

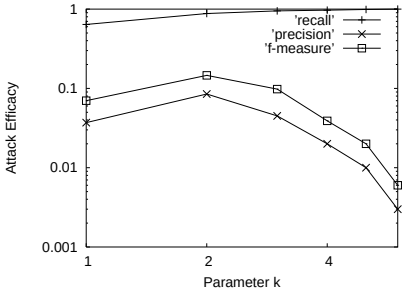

Fig. 11. Statistical Shortest Path Tracing: 128 Erlang Call Volume

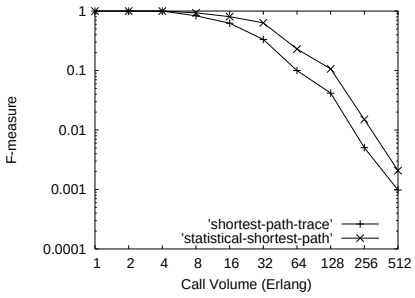

Fig. 12. F-measure with κ = 2

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:

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

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 $f(p) > 0$. Consistent with other work in this research area, 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.

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 $f(p_1)$, $f(p_2)$, $\cdots$ $f(p_N)$ in that order.

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 $M_{ij} = \frac{nf(p_i \rightarrow p_j)}{in(p_i)}$ if there exists a directed edge $p_i \rightarrow p_j$ 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 iteratively: $\pi^{t+1} = \pi^t M$ starting with π_i^0 = 1 if $p_i = src$; π_i^0 = 0 otherwise. Assuming M is irreducible, π converges to a steady state solution in $O(N \log N)$ iterations.

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 $p \rightarrow q$ is $\frac{nf(p \rightarrow q)}{in(p)} * Pr(hop \geq hc(src, p) + 1 \mid hop \geq hc(src, p))$, where $hc(src, p)$ denotes the number of hops along the shortest path between src and p on graph G^κ. $Pr(hop \geq hc(src, p) + 1 \mid hop \geq hc(src, p)) = \frac{Pr(hop \geq hc(src,p)+1)}{Pr(hop \geq hc(src,p))}$ 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.

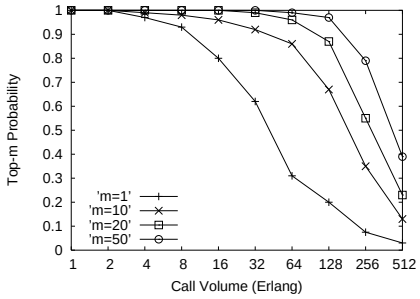

Fig. 13. Top-m Probability with κ = 2 and Distance prior

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:

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

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

In our enterprise data set (see Section II), src and dst were chosen randomly from the network. Hence we compute hop count and distance distribution using randomly chosen pairs of src and dst.

Figure 13 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.

III. Related Work

Mix [8] 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 [8][21]. Onion routing [13] and its second generation Tor [9] aim at providing anonymous transport of TCP flows over the Internet. ISDN mixes [17] 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[5].

It is widely acknowledged that low latency anonymizing networks [9][13][6] are vulnerable to timing analysis attacks [23][21], especially from well placed malicious attackers [28]. Several papers have addressed the problem of tracing encrypted traffic using timing analysis [27][29][31][26][7][10][25]. 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.

Tarzan [12] 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 [18][20]. 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.

IV. 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. We have presented detailed experimental evaluation that demonstrates the efficacy of these attacks in a VoIP network.

[5]According to TeleGeography Research [5], world wide VoIP's share of voice traffic has grown from 12.8% in 2003 to an estimated 75% 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] Skype - the global internet telephone company. http://www.skype.com.

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

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

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

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

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

[10] 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.

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

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

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

[14] 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.

[15] J. Y. Hui. Switching and traffic theory for integrated broadband networks. Academic Press ISBN: 079239061X, 1990.

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

[17] 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.

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

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

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

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

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

[23] 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.

[24] C. J. Van-Rijsbergen. Information retrieval. In *Second Edition, Butterworths, London*, 1979.

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

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

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

[28] 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.

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

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

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