Finding "Who Is Talking to Whom" in VoIP Networks via Progressive Stream Clustering

Olivier Verscheure [†] Michail Vlachos [†] Aris Anagnostopoulos [‡]
Pascal Frossard [⋆] Eric Bouillet [†] Philip S. Yu [†]

[†] IBM T.J. Watson Research Center
[‡] Yahoo! Research
[⋆] EPFL

Abstract

Technologies that use the Internet network to deliver voice communications have the potential to reduce costs and improve access to communications services around the world. However, these new technologies pose several challenges in terms of confidentiality of the conversations and anonymity of the conversing parties. Call authentication and encryption techniques provide a way to protect confidentiality, while anonymity is typically preserved by an anonymizing service (anonymous call).

This work studies the feasibility of revealing pairs of anonymous and encrypted conversing parties (caller/callee pair of streams) by exploiting the vulnerabilities inherent to VoIP systems. In particular, by exploiting the aperiodic inter-departure time of VoIP packets, we can trivialize each VoIP stream into a binary time-series. We first define a simple yet intuitive metric to gauge the correlation between two VoIP binary streams. Then we propose an effective technique that progressively pairs conversing parties with high accuracy and in a limited amount of time. Our metric and method are justified analytically and validated by experiments on a very large standard corpus of conversational speech. We obtain impressively high pairing accuracy that reaches 97% after 5 minutes of voice conversations.

1 Introduction

The International Telecommunications industry is in the early stages of a migration to Voice over Internet Protocol (VoIP). VoIP is a technology that enables the routing of voice conversations over any IP-based networks such as the public Internet. The voice data flows over a general-purpose packet-switched network rather than over the traditional circuit-switched Public Switched Telephone Network (PSTN). Market research firms including In-Stat and IDC predict that 2005-2009 will be the consumer and small business VoIP ramp-up period, and migration to VoIP will peak in the 2010-2014 time frame. Research organization Gartner Inc. recently reported that spending by U.S. companies and public-sector organizations on VoIP systems is on track to grow to \$903 million in 2005 (up from the \$686 million in 2004). Gartner expects that by 2007, 97% of new phone systems installed in North America to be VoIP or hybrids.

While the migration to VoIP seems inevitable, there are security risks associated with this technology that are carefully being addressed. Eavesdropping is one of the most common threats in a VoIP environment. Unauthorized interception of audio streams and decoding of signaling messages can enable the eavesdropper to tap audio streams in an unsecured VoIP environment. Call authentication and encryption mechanisms [2, 15] are being deployed to preserve customers' confidentiality. Preserving customers' anonymity is also crucial, which encompasses both the identity of the people involved in a conversation and the relationship caller/callee (pair of voice streams). Anonymizing overlay networks such as Onion Routing [7] and FindNot.com [16] aim at providing an answer to this problem by concealing the IP addresses of the conversing parties. A recent work [14] shows that tracking anonymous peer-to-peer VoIP calls on the Internet is actually feasible. The key idea consists in embedding a unique watermark into the encrypted VoIP flows of interest by minimally modifying the departure time of selected packets. This technique transparently compromises the identity of the conversing parties. However, the authors rely on the strong assumption that one has access to the customer's communication device, so that the watermark can be inserted before the streams of interest reach the Internet.

This work studies the feasibility of revealing pairs of *anonymous* conversing parties (caller/callee pair of streams) by exploiting the vulnerabilities inherent in VoIP systems. Using the methods provided in this work, we also note one seemingly surprising result; that the proposed techniques are applicable even when the voice packets are encrypted. While the focus of this work is on VoIP data, the techniques presented here are of independent interest and can be used for pairing/clustering any type of binary streaming data. The contributions of the paper function on different

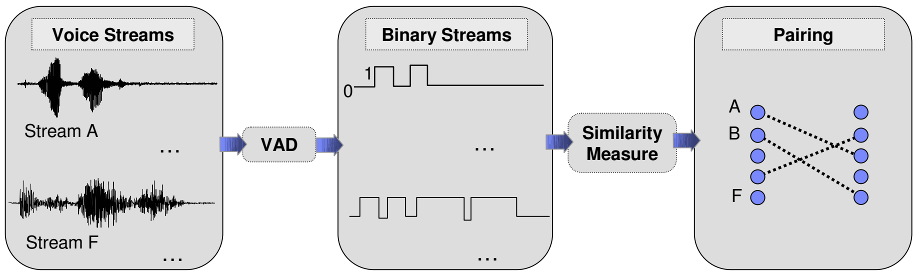

Figure 1: Overview of the proposed methodology

levels:

1. We formulate the problem of pairing anonymous and encrypted VoIP calls.
2. We present an elegant and fast solution for the conversation pairing problem, which exploits well established notions of complementary speech patterns in conversational dynamics.
3. Our solution is based on an efficient transformation of the voice streams into binary sequences, followed by a progressive clustering procedure.
4. Finally, we verify the accuracy of the proposed solution on the very large dataset of voice conversations.

The paper is organized as follows. Sections 2 - 5 present our solution for pairing conversations over any medium by mapping the problem into a complementary clustering problem for binary streaming data. We introduce various intuitive metrics to gauge the correlation between two binary voice streams and we present effective methods for progressively pairing conversing parties with high accuracy within a limited amount of time. Section 6 shows how the presented solution can be adapted for a VoIP framework, and demonstrates that encryption schemes do not hinder the applicability of our approach. Section 7 validates the presented algorithm by experiments on a very large standard corpus of conversational speech [8]. Finally, we provide our concluding remarks in Section 8 and we also instigate directions for future work.

2 Problem Overview and Methodology

We start with a generic description of the conversation pairing problem, with the intention of highlighting the key insights governing our solution. We will later clarify the required changes so as the following model can be adapted for the VoIP scenario.

2.1 Pairing Voice Conversations

Let us assume that we are monitoring a set $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of k voice streams (for now suppose k is even), comprising a total of $k/2$ conversations[1]. Each stream holds one-way of a two-way voice conversation, and there exists also a homologue voice stream that holds the other side of the conversation. Our objective is to efficiently reveal the relationship of *two-way* conversing parties. For example, assume S_2 is actually involved in a conversation with S_5. We aim at finding all relationships $S_i \leftrightarrow S_j$ including the example $S_2 \leftrightarrow S_5$, such that streams S_i and S_j correspond to each one-way voice stream of the same conversation. Our approach does not require an even number of voice streams and we also do not assume each voice stream to have a matching pair. Any voice stream without a corresponding counterpart is referenced to henceforth as a *singleton* stream. At the end of the pairing process some streams may remain unmatched. These will be the voice streams for which the algorithm either does not have adequate data to identify a match, or is not in a position to discriminate with high confidence a conversational pair.

The key intuition behind our approach is that conversing parties tend to follow a "complementary" speech pattern. When one speaks, the other listens. This "turn-taking" of conversation [13] represents a basic rule of communication, well-established in the fields of psycholinguistics, discourse analysis and conversation analysis, and it also manifests under the term of speech "coordination" [4]. Needless to say, one does not expect a conversation to follow strictly the aforementioned rule. A conversational speech may well include portions where both contributers speak or are silent. Such situations are indeed expected, but in practice they do not significantly "pollute" the results, since given conversations of adequate length, coordinated speech patterns are bound to dominate. We will show this more explicitly in the experimental section, where the robustness of the proposed measures are tested also under conditions of network latency.

Using the above intuitions, we will follow the subsequent steps for recognizing pairs of conversations:

1. First, voice streams are converted into binary streams, indicating the presence of voice (1) and silence (0).
2. Second, we leverage the power of complementary similarity measures, for quantifying the degree of coordination between pairs of streams.
3. Using the derived complementary similarity, we will employ a progressive clustering approach for deducing conversational pairs.

[1]In this paper we will not deal with multi-way conversations.

A schematic of the above steps is given in Figure 1. In the following section we will first place our approach within the context of related work.

2.2 Related Work

Recent work that studies certain VoIP vulnerabilities and has attracted a lot of media attention has appeared in [14]. The authors present techniques for watermarking VoIP traffic, with the purpose of tracking the marked VoIP packets. For accomplishing that, however, initial access to a user's device or computer is required. In this work we achieve a different goal; that of identifying conversational pairs, however we do not assume any access to a user's device. The only requirement of our approach (more explanations will be provided later) is the provision of a limited number of network *sniffers*, which will capture the incoming (encrypted) VoIP traffic.

Relevant to our approach are also recent techniques for clustering binary streams [11, 10]. These consider clusters of objects and not pairs of streams, which is one of the core requirements for the application that we examine. The algorithm presented in this work has the additional advantage of being *progressive* in nature, returning identified pairs of streams before the complete execution of the algorithm. In [6], Cormode et al., study the use of binary similarity measures for comparison streams, and focus on sketch approximations of the Hamming Norm. This work examines the use of *complementary* binary similarity measures between streaming data.

The methods presented in this work, exploit and adapt data-mining techniques for depicting inherent vulnerabilities in VoIP streams, which can potentially compromise the users' anonymity. It is interesting to note, that much of recent work in data-mining [9, 5] has focused on how to embed or maintain privacy for various data-mining techniques, such as clustering, classification, and so on.

In the sections that follow we will provide a concise description of a Voice Activity Detector. We will also present intuitive *coordination* measures for quantifying the complementary similarity between binary streams. We put forward a lightweight pairing technique based on adaptive soft decisions for reduce the pairing errors and avoiding the pairing of singleton streams. Key requirements include lightweight processing, quick and accurate identification of the relationships, and resilience to both noise and latency.

3 Voice Activity Detection

The goal of a Voice Activity Detection (VAD) algorithm is to discriminate between voiced versus unvoiced sections of a speech stream. We provide only a high-level description of a typical VAD algorithm for reasons of completeness, since it is not the focus of the current work. The VAD process computes the energy of small overlapping speech packets (also called *frames*, with each frame being 20-30msec in length), and employs an adaptive energy threshold that will differentiate the voiced from the unvoiced frames. The threshold is typically deduced by estimating the average energy of the unvoiced portions, taking also into consideration a background noise model, based on the characteristics of the data channel. The output of the VAD algorithm will be "1" when there is speech detected and "0" in the presence of silence. A simple schematic of its operation is provided in Fig. 2.

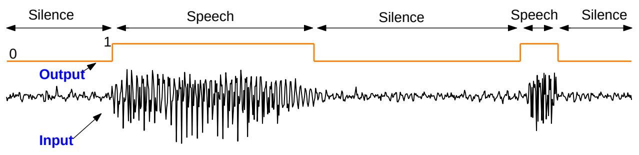

Figure 2: A Voice Activity Detector can effectively recognize the portions of silence or speech on a voice stream.

As will be explained later, the voice activity detection is inherently provided by the VoIP protocol.

4 Coordination Measures

After voice activity detection is performed, each voice stream S_i is converted into a binary stream B_i. The resulting binary stream only holds the necessary information that indicates the speech/no-speech patterns. The objective now is to quantify the *complementary similarity* (which we call *cimilarity*) between two binary streams.

As already mentioned, the basic insight behind detecting conversational pairs is to discern voice streams that exhibit complementary speech behavior. That is, given a large number of binary streams $B_1, B_2, \ldots, B_N$, and a query stream B_q (which indicates the voice activity of user q), we would like to identify the stream B_j that is most complementary similar to stream B_q, or in other words, has the largest cimilarity.

We present different versions of cimilarity measures (*Cim*) and we later quantify their performance in the experimental section. Let us consider two binary streams, B_i and B_j. By abstracting B_i and B_j as binary sets, an intuitive measure of coordination between users i and j consists in computing the intersection between B_i and the binary complement of B_j normalized by their union. We denote by *Cim-asym*(i, j, T) this measure computed over streams B_i and B_j after T units of time. One can readily verify that it can be written as:

$$\text{Cim-asym}(i, j, T) = \frac{\sum_{t=1}^{T} B_i[t] \wedge \neg B_j[t]}{\sum_{t=1}^{T} B_i[t] \vee \neg B_j[t]}. \qquad (1)$$

where $B_k(t) \in \{0, 1\}$ is the binary value for user k at time t, and the symbols $\wedge$, $\vee$, and $\neg$ denote the binary AND, OR and NOT operators, respectively.

Note that Equation 1 asymmetrically measures the amount of coordination between speakers i and j. That is, in general, *Cim-asym*$(i, j, T) \neq$ *Cim-asym*(j, i, T) due to the binary complement operator. This measure can be seen as the asymmetric extension of the well-known Jaccard coefficient [3]. Thus, we also refer to this measure as *Jaccard-Asymmetric*.

$B_i \setminus B_j$	1	0
1	0	1
0	0	0

$B_i \setminus B_j$	1	0
1	1	1
0	0	1

Figure 3: Computation of `Cim-asym`, *Left:* Numerator, *Right:* Denominator

Computing `Cim-asym` between two binary streams is computationally very light. The computation lookup table for the numerator and the denominator is provided in Figure 3. The numerator is increased when $B_i = 1$ and $B_j = 0$, while the denominator is not increased when $B_i = 0$ and $B_j = 1$. So *Cim-asym*(B_i, B_j) only rewards the presence of non-speech of user j, when user i speaks.

Example: Given $B_1 = 11100110$ and $B_2 = 00010001$ then *cim-asym*(B_1, B_2) = 0.833 and *cim-asym*(B_2, B_1) = 0.6667.

The `Cim-asym` measure is also easily amenable to incremental maintenance as time T progresses. Indeed, let $V_\wedge(i, j)$ and $V_\vee(i, j)$ denote the running values of the numerator and the denominator, respectively. The value of *Cim-asym*(i, j, T) for any elapsed time T is given by the ratio $V_\wedge(i, j)/V_\vee(i, j)$. Therefore, given n binary streams, incrementally computing *Cim-asym* requires keeping 2 times $n(n-1)$ values in memory. For example, when monitoring $n = 1000$ streams and assuming each value is stored as an *int16*, only 4 MBytes of memory are needed for tracking all the required statistics.

We also consider a symmetric extension of *Cim-asym* denoted by *Cim-sym* and referred to as *Jaccard-Symmetric*. This intuitive extension is written as:

$$\text{Cim-sym}(i, j, T) = \sum_{t=1}^{T} \frac{(B_i[t] \wedge \neg B_j[t]) \vee (\neg B_i[t] \wedge B_j[t]))}{T} = \sum_{t=1}^{T} \frac{\text{XOR}(B_i[t], B_j[t])}{T}, \qquad (2)$$

This metric is even simpler than its asymmetric version. Moreover, given n binary streams, incrementally computing *Cim-sym* requires keeping only $\binom{n}{2}$ values in memory thanks to its symmetric nature. Using the example above, memory requirements drop to approximately 1 Mbytes given the same assumptions. The *Cim-sym* is generally more aggressive than its asymmetric counterpart, because it also rewards the presence of speech patterns when the user in question does not speak. However, our experiments indicate that the most conservative asymmetric version ultimately achieves the best detection accuracy.

Finally we consider the *Mutual Information* (MI) as a measure of coordination between conversing parties [1]. This is a measure of how much information can be obtained about one random variable B_i by observing another B_j. Let $p_{i,j}(x, y)$, $p_i(x)$, and $p_j(y)$ with $x, y \in 0, 1$ denote the joint and marginal running averages for users i and j after T units of time. For example,

$$p_{i,j}(0, 1) = \frac{1}{T} \sum_{t=1}^{T} \neg B_i[t] \wedge B_j[t].$$

The amount of *Mutual Information* (MI) between streams B_i and B_j is written as:

$$\text{MI} = \sum_{x,y \in 0,1} p_{i,j}(x, y) \log_2 \frac{p_{i,j}(x, y)}{p_i(x) p_j(y)} \qquad (3)$$

The mutual information measure requires higher processing power but exhibits symmetry. Note that while at first it seems that one needs to store 8 statistics for updating the Mutual Information, in fact only 3 statistics are required. For example $p_{i,j}(0, 0)$, $p_{i,j}(0, 1)$ and $p_{i,j}(1, 0)$ are sufficient to restore the remaining ones, since:

$$p_{i,j}(1, 1) = 1 - p_{i,j}(0, 0) - p_{i,j}(0, 1) - p_{i,j}(1, 0)$$
$$p_i(0) = p_{i,j}(0, 0) + p_{i,j}(0, 1)$$
$$p_j(0) = p_{i,j}(0, 0) + p_{i,j}(1, 0)$$

and so on.

So, given n binary streams, it can be shown that incrementally computing *MI* requires keeping 3 times $\binom{n}{2}$ values in memory thanks to its symmetric nature. Thus, approximately 3 MBytes of memory are required for the above example.

In the following section, we illustrate how any of the above metrics can be used in conjunction with a progressive clustering algorithm for identifying conversing pairs.

5 Conversation Pairing/Clustering

In order to get insights about the pairing algorithm we first plot how the complementary similarity of one voice stream progresses over time against all other streams (Figure 4). Similar behavior is observed for the majority of voice streams.

One can notice that voice pairing is extremely ambivalent during the initial stages of a conversation, but the uncertainty decreases as conversations progress. This is observed, first, because most conversations in the beginning

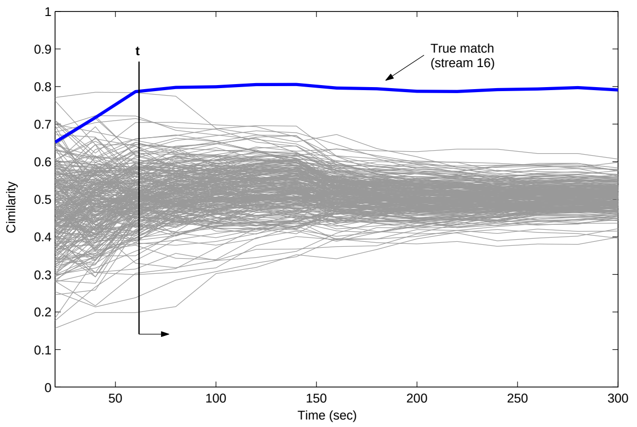

Figure 4: Progression of complementary similarity (*Jaccard-Asymmetric*) over time for stream 1, against all other voice streams. The true match has the highest value after time t.

exhibit a customary dialog pattern ("hi," "how are you," etc.). However, conversations are bound to evolve in different conversational patterns, leading to a progressive decay in the matching ambiguity. Second, some time is required to elapse, so as the Law of Large Numbers can come to effect.

A simple solution for tackling the conversation pairing problem would be to compute the pairwise similarity matrix M after some time T, where each entry provides the complementary similarity between two streams:

$$M(i,j) = \text{Cim}(i,j,T),$$

where Cim is one of the cimilarity measures that we presented in Section 4.

Then we can pair users i and j if we have

$$M(i,j) = \max_{\ell}\{M(i,\ell)\}$$

and

$$M(i,j) = \max_{\ell}\{M(\ell,j)\}.$$

We call this approach *hard* clustering, because at each time instance it provides a rigid assignment of pairs, without providing any hints about the confidence or ambiguity of the matching.

There are several shortcomings that can be identified with the above hard clustering approach:

- First, it provides no concrete indication when the pairing should start. When are the sufficient statistics robust enough to indicate that pairing should commence?

- In order to achieve high accuracy, sufficient data need to be collected. This penalizes the system responsiveness (no decision is made until then) and additionally significant resources are wasted (memory and CPU).

- Different streams will converge at different rates to their expected similarity value. Therefore, decisions for different pairs of streams can (or cannot) be made at different times, which is not exploited by the hard clustering approach.

In the stream-pairing algorithm that we describe below, we will address all the previous issues, allowing the early pairing of streams, while imposing minimal impact on the system resources.

Using as a guide the aforementioned behavior which governs the progression of cimilarity, we construct the clustering algorithm as an *outlier detection scheme*. What we have to examine is whether the closest match is "sufficiently distant" from the majority of streams. Therefore, when comparing a stream (e.g. stream 1) against all others, the most likely matching candidate should not only hold the maximum cimilarity, but also deviate sufficiently from the cimilarity of the remaining streams.

```
1.  Function matchStreams($\mathcal{S}$, $f$)
2.    /* $\mathcal{S}$ contains all the streams $[1:N]$ */
3.    for ($t = 0, 1, 2, \ldots, T$)
4.    /* $T$ is an upper bound that will depend on $n$ (Ideally,
       $T = \Theta(\ln n)$) */
5.      Update the pairwise similarity matrix $M(\cdot,\cdot)$
6.      foreach unmatched stream $s_i$
7.        compute $\max_1$, $\max_2$ of $M(s_i,\cdot)$
8.        $sm_i$ ← stream ID of $\max_1$
9.        trimmedMsi ← $k$-trim of $M(s_i,\cdot)$
10.       cMass ← mean of trimmedMsi
11.       if ($\max_1 - \max_2 > f \cdot (\max_2 - \text{cMass})$)
12.       /* stream $s_i$ matches with stream $sm_i$ */
13.         remove rows $s_i$, $sm_i$ from $M$
14.         remove columns $s_i$, $sm_i$ from $M$
15.     if (all streams are paired) return
```

Figure 5: Progressive algorithm for matching streams.

Figure 5 contains a pseudocode of the pairing algorithm, while Figure 6 depicts the steps behind its execution. We maintain the same matrix M as in the hard-clustering approach, which is updated as time progresses. Then, at every step of the algorithm, for every stream that has not been matched, we perform the following actions. Suppose that at any time T we start with the binary stream B_1:

1. We perform a k-Trim for removing the k most distant and k closest matches (typically $k = 2 \ldots 5$).

2. We compute the average cMass (center of mass) of the remaining stream cimilarities.

3. We record the cimilarity of the two closest matches to stream B_1, which we denote as $\max_1$ and $\max_2$. We consider the closest match "sufficiently separated" from the remaining streams if the following holds:
$$\max_1 - \max_2 > f \cdot (\max_2 - \text{cMass}),$$
where the f constant captures the assurance (confidence) about the quality of our match (values of f range within $0.5 \ldots 2$). Greater values of f, signify more separable best match compared to the remaining streams, and hence more confident matching.

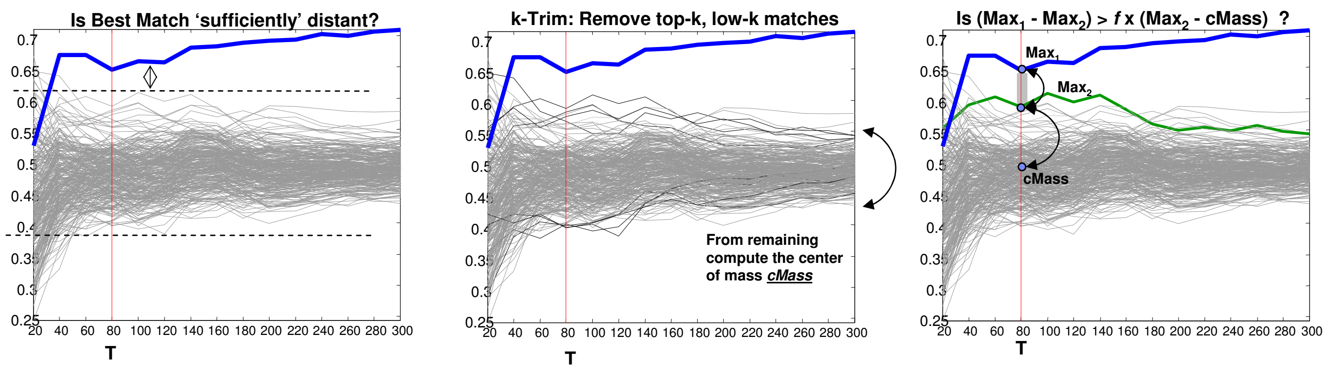

Figure 6: Pairing of voice conversations.

4. If the above criterion does not hold we cannot make a decision about stream B_1, otherwise we match B_1 with $B_{\max_1}$ and we remove their corresponding rows and columns from the pairwise cimilarity matrix (Figure 7)

Notice that the *outlier detection criterion* adapts according to the current similarity distribution, being more strict in the initial phases (wider $(\max_2 - \text{cMass})$) and becoming more flexible as time passes.

Furthermore, the algorithm does not need to know a priori a bound on the required time steps for execution. As soon as there is sufficient information, it makes use of it and it identifies the likely pairs. In the next section we analyze the performance of the algorithm.

5.1 Time and Space Complexity

Compared to the hard-clustering approach that needs to recompute the pairwise similarity matrix M for every time step, the progressive algorithm reduces the computational cost by progressively removing from the distance computation the streams that have already been paired, although the initial stages of the algorithm are somehow more expensive, since in every iteration there are more operations performed than just the update of the similarity matrix M.

So, let us analyze the time and space that our algorithm requires. Initially there are n streams, so the size of matrix M is n^2, hence the space requirement is $O(n^2)$.

For the time complexity, assume that at time step t there are S_t streams available. Then the running time required to execute the t^{th} step is $O(S_t^2)$. To see that, notice that line 5, where we update the cimilarity matrix M, requires $O(S_t^2)$ time steps. The **foreach** loop at line 6, where we process each stream, is executed at most S_t times and each of the commands inside the loop can be computed in linear (in S_t) time. (The most involved is line 9 for computing the k-trim, which can be done with a variation of a linear algorithm for computing the median.) Therefore, the running time of every time step of the algorithm is $O(S_t^2)$, in other words there exists a constant κ such that the time per step is bounded by $\kappa \cdot S_t^2$.

Therefore, if T_r is the total running time, we have

$$T_r \le \kappa \cdot \sum_{t=1}^{\infty} S_t^2,$$

where κ is the constant hidden in the asymptotic notation. It is therefore clear that in order to analyze the running time we have to evaluate how the values S_t decrease, and in particular when S_t becomes 0. Our experimental results, depicted in Figure 15, indicate that a candidate function for S_t is given by the sigmoidal function

$$S_t = n \cdot \frac{e^{-\frac{t-c_1 \ln n}{c_2}}}{1 + e^{-\frac{t-c_1 \ln n}{c_2}}},$$

for some constants c_1 and c_2, which in our case we estimated as 15 and 20, respectively. In Figure 8 we show the sigmoidal function that matches the observed data.

Having this in mind, we can estimate

$$\begin{aligned}
\sum_{t=1}^{\infty} S_t^2 &= \sum_{t=1}^{\infty} n^2 \cdot \frac{e^{-\frac{2(t-\ln n)}{c_2}}}{\left(1 + e^{-\frac{t-c_1 \ln n}{c_2}}\right)^2} \\
&= \sum_{t=1}^{c_1 \ln n} n^2 \cdot \frac{e^{-\frac{2(t-\ln n)}{c_2}}}{\left(1 + e^{-\frac{t-c_1 \ln n}{c_2}}\right)^2} \\
&\quad + \sum_{t=c_1 \ln n+1}^{\infty} n^2 \cdot \frac{e^{-\frac{2(t-\ln n)}{c_2}}}{\left(1 + e^{-\frac{t-c_1 \ln n}{c_2}}\right)^2} \\
&\le c_1 \cdot n^2 \ln n + O(n^2),
\end{aligned}$$

and, therefore, $T_r \le c_1 \cdot \kappa \cdot n^2 \ln n + O(n^2)$. Notice that we can obtain a similar bound (with worse constants) by noticing (after some calculations) that after time $t = (c_1 + c_2) \ln n$ we have $S_t < 1$, so we can bound the running time by $(c_1 + c_2)\kappa \cdot n^2 \ln n$.

Therefore, the algorithm is efficient, since it only introduces an $O(n \log n)$ complexity per data stream by progressively removing paired matches. In Figure 9 we depict

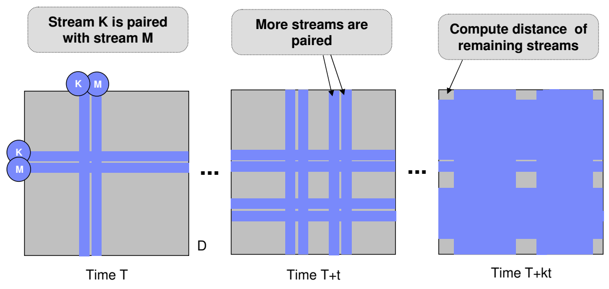
Stream K is paired with stream M
More streams are paired
Compute distance of remaining streams
K
M
D
Time T
Time T+t
Time T+kt

Figure 7: The similarity/dissimilarity matrix does not have to be completed fully, for all time instances. Matching pairs can be removed from computation.

the lifecycle of various streams for an experiment with 280 voice streams. Each line represents a voice stream and is extended only up to the point that the stream is paired.

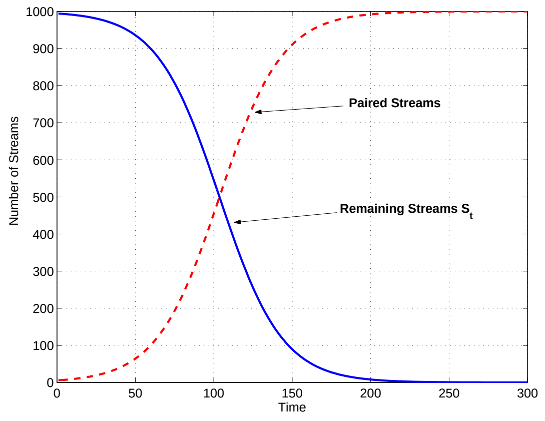

Figure 8: A sigmoidal function that models the results in Figure 15.

Note that this analysis is based on our observed data. A more rigorous approach can consider some underlying probability space to model the streams generations. Then T_r and S_t are random variables and we can study quantities such as the expected running time and variance, or give large-deviation bounds. For example, for the expected running time of the algorithm, we have

$$\mathrm{E}[T_r] \leq \mathrm{E}\left[\kappa \cdot \sum_{t=1}^{\infty} S_t^2\right] = \kappa \cdot \sum_{t=1}^{\infty} \mathrm{E}\left[S_t^2\right],$$

since the S_t's are nonnegative. This gives the interesting conclusion that the expected running time depends on the variance of the number of streams that remain unpaired throughout the execution of the algorithm.

6 Extending to a VoIP network

We explain how the previous model of pairing voice conversations can be extended to work on a voice-over-IP network. In what follows we describe the structure and transmission protocol of a typical VoIP network and we illustrate

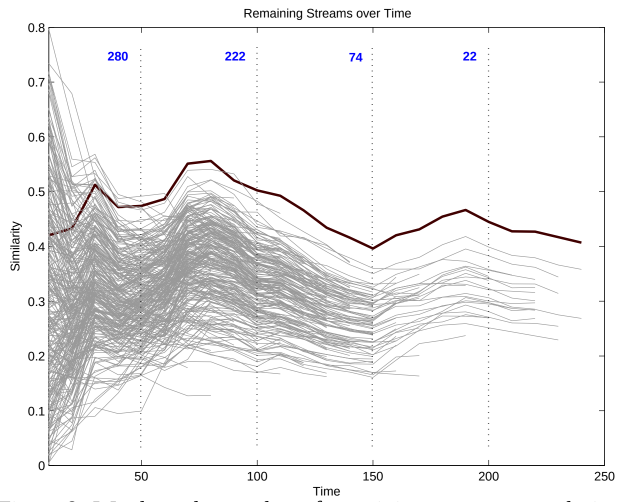

Figure 9: We show the number of remaining streams at each time instance on an experiment with 280 streams. The darker stream indicates the actual best match which is identified after 240 seconds.

the steps for reconstructing the binary voice activity stream from a sequence of VoIP packets.

We consider the framework depicted in Figure 10. N VoIP subscribers are connected to the Internet either directly via their ISP providers, or behind VoIP gateways on traditional PSTN networks. Those VoIP subscribers may use a low-latency anonymizing service composed of a set of overlay network nodes. Each VoIP stream traverses a possibly distinct set of IP routers, a subset of which are assumed to have VoIP sniffing capabilities. Each sniffer preprocesses the incoming VoIP traffic and forwards the resulting data to a central processing unit.

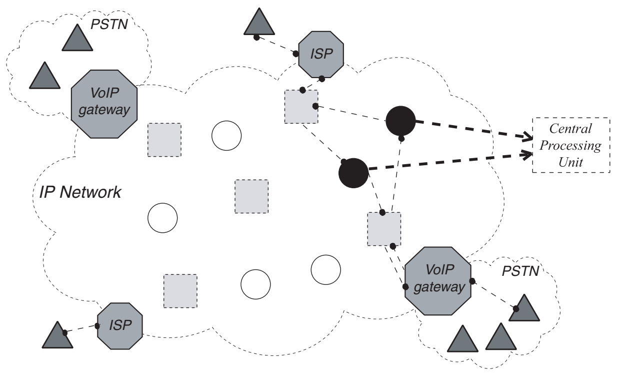

Figure 10: VoIP Framework. A set of customers (triangles) with direct access to the IP network or behind a PSTN network. A set of IP routers (white circles), a subset of which are VoIP sniffers (black circles) that forward preprocessed VoIP data to a Central Processing Unit. An anonymizing network composed of a set of overlay nodes (light-shaded squares).

A voice signal captured by a communication device goes through a series of steps in preparation for streaming. Figure 11 summarizes some of the following concepts. A voice signal is continually captured by the microphone of a communication device. The digital signal is segmented and fed to a Voice Activity Detection (VAD) unit. This feature allows VoIP devices to detect whether the user is currently speaking or not by analyzing voice activity. Whenever the voice activity is below a certain adaptive threshold, the current segment is dropped. Note that if the VAD algorithm is not sophisticated enough, actual voiced segments may get wrongly filtered out [12]. The filtered signal is then passed through a voice codec unit (e.g., G.729.1 or GSM) that compresses the input voice segments to an average bit rate of approximately 10 Kbps. Those compressed segments are encrypted using 256-bit AES [2] and packetized using the Real-time Transport Protocol (RTP). Each RTP packet consists of a 12-byte header followed by 20ms worth of encrypted and compressed voice. It is important to note that all RTP headers are in the clear [2]. Various RTP header fields are of great interest for our purpose. In particular, the Payload Type (PT) field enables easy spotting of VoIP streams, the Synchronization Source (SSRC) field uniquely identifies the stream of packets, and the Timestamp field reflects the sampling instant of the first byte in the RTP payload. Finally, each RTP packet is written to a network socket.

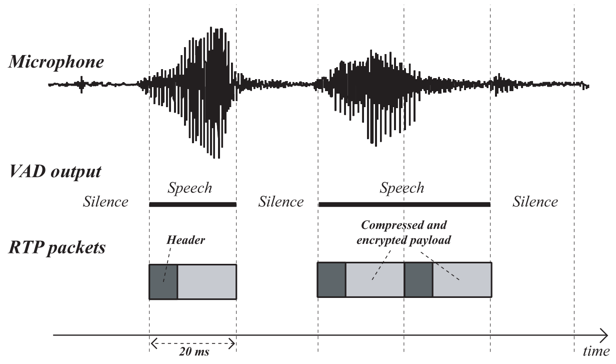

Figure 11: A voice signal captured by a communication device goes through various steps in preparation for streaming.

6.1 Separating the voice streams

For recasting the problem into the scenario that we previously studied, we need first to reconstruct the binary streams indicating the voice activity of each one-way communication. Given the above transmission protocol a VoIP sniffer that gathers incoming internet traffic can identify and separate the different voice streams and also convert them into binary streams that indicate periods of activity or silence as follows:

1. The RTP PT field is used to segregate VoIP packets from different data traffic (see Figure 12)
2. Each different voice stream can be tracked by its unique RTP SSRC field.

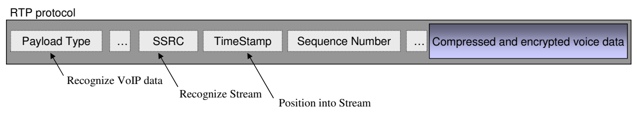

Figure 12: Fields of the RTP protocol that are used

3. Finally, the binary stream indicating the presence of speech or silence, is inherently provided by the VoIP protocol, given the presence or not of a voice packet. Packets are only sent during speech activity which constitutes an indirect way of constructing the binary voice activity stream. For a given RTP SSRC value (stream ID), a sniffer measures the difference of two consecutive Timestamp values (inter-departure time of packets) and generates a one (or a zero) if the difference is equal to (or larger than) the segmentation interval of 20ms. Thus, each binary stream results from the aperiodic inter-departure time of VoIP packets derived from the Voice Activity Detection (VAD), which is performed within the customer's communication device. In Figure 13 we visualize this process.

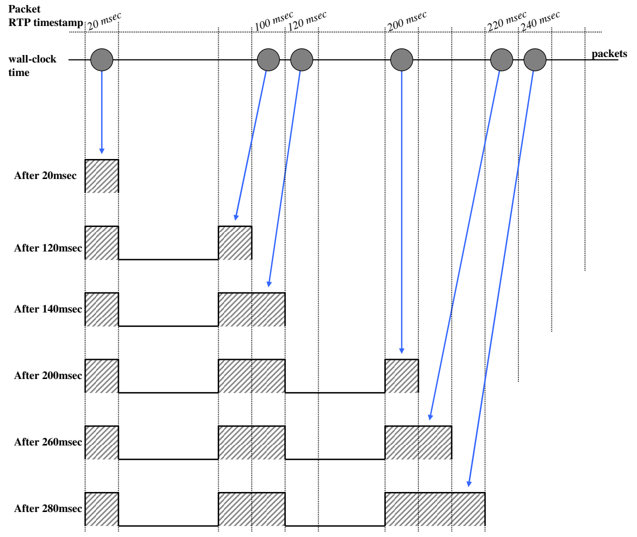

Figure 13: Usage of RTP Timestamp for reconstructing the binary voice activity sequence

6.2 Advantages and Discussions

Several are the advantages of the presented approach:

- A very important first outcome of the presented approach is that it operates on the compressed data domain. Because we do not need to decompress the voice data to perform any action, this immediately gives a significant performance advantage to our approach.
- A surprising second observation, is that the presented methodology is also valid even when the voice data is

encrypted! This is true because for performing voice activity detection we merely exploit the presence of the packet as an indication of speech. Note, that because the data can still be encrypted, the privacy of the conversation content is not violated.

- Finally, the presented algorithm is very robust to jitter and network latency. Jitter has no effect on the RTP Timestamp values, which are assigned during the data transmission and measure the *inter-departure* packet time. Network latency only affects the arrival of the first packet, since synchronization of subsequent packets can be reconstructed by the corresponding RTP timestamps. In our experiments we do not assume a zero-latency network. Instead we show that our method is indeed resilient to latency.

Lastly, we briefly elaborate on certain issues or questions that may arise given the dynamic nature of the system:

a) We do not assume that the network sniffers are able to track all voice streams. Singleton streams can be present. This does not pose a problem for our algorithm since we do not enforce pairing of all streams.

b) The cardinality of voice conversations captured by the sniffer changes over time as calls start and/or terminate. Therefore, one should pair streams that commence at approximately the same time (within twice the assumed worse network latency). This gives rise to multiple cimilarity arrays formed by voice streams with similar arrival times. In the experiments we do not consider this scenario, but we experiment with k voice streams that begin simultaneously for illustrating better the scalability and accuracy of our approach under the maximum possible load.

c) Finally, it is worth noting that the RTP Sequence Number field together with the Timestamp values help avoid blindly concluding a packet has been filtered out by the VAD unit while, in fact, it has been dropped by the network. However losses are seldom in commercial VoIP networks and in this work we do assume a lossless VoIP framework.

7 Experiments

As our experimental testbed we used real telephony conversations from switchboard data [8], which contained 500 pairs of conversations for a total of 1000 voice streams and consisted of multiple pairs of users conversing on diverse topics. Such datasets are typically used in many speech recognition contests for quantifying the quality of different speech-to-text processes. The specific dataset that we used, consisted actually of quite noisy conversational data and the length of each conversation is 300 sec. The original voice data have been converted to VoIP packets (using the protocol described in the VoIP section), then fed onto a local network using our custom made workload generator and recaptured by the data sniffers.

7.1 Comparison of Cimilarity Measures

In this initial experiment we compare the pairing accuracy of the three presented complementary similarity measures. We utilize the hard clustering approach which does not leave any unassigned pairs, therefore it introduces a larger amount of incorrectly classified pairs. However, since the hard clustering follows a more aggressive pairing strategy, this experiment essentially showcases the best possible convergence rate of the various similarity measures.

Figure 14 presents the pairing accuracy of the Mutual Information (MI), Jaccard Asymmetric and Jaccard Symmetric measures. Every 10 seconds we calculate the recognition accuracy by pairing each of the voice streams with the stream that depicts the maximum complementary similarity. Notice than in this way we do not necessarily impose a 1-to-1 mapping of the streams (hence, a stream may be paired with more than one streams). We report the results using the 1-to-n mapping, since we discovered that it consistently achieves more accurate results than the 1-to-1 mapping.

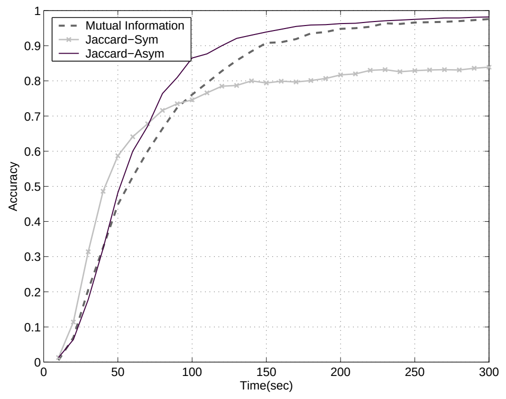

Figure 14: Pairing accuracy between 3 measures.

On the figure we can observe that the Asymmetric-Jaccard measure is the best overall performer. It achieves faster convergence rate than the Mutual Information (90% accuracy after 120sec, instead of 150sec for the MI) and also a larger amount of correctly classified pairs at the end of the experiment. The Symmetric-Jaccard measure appears to be quite aggressive in its pairing decisions in the beginning, but flattens out fairly quickly, therefore it cannot compete in terms of accuracy with the other two measures. Since different measures appear to exhibit diverse convergence rates, as possible future work it would be interesting to explore the possibility of alternating use for the various measures at different stages of the execution, in order to achieve even faster pairing decisions.

In general, the results of this first experiment are very encouraging, since they indicate that the use of simple matching measures (like the Asymmetric Jaccard) can achieve comparable or better pairing accuracy than more complex measures (such as the Mutual Information). For the remainder of the experiments we will focus on the Asymmetric-Jaccard measure, and specifically on its performance using the progressive pairing algorithm.

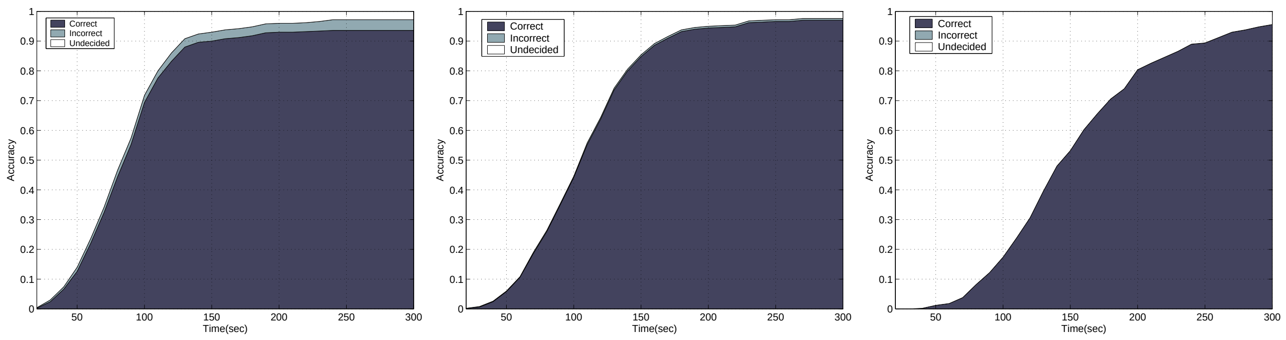

Figure 15: Progressive pairing for Asymmetric Jaccard. Left: $f = 1/2$, Middle: $f = 2/3$, Right: $f = 1$

7.2 Progressive Clustering Accuracy

The progressive algorithm presented in the paper has two distinct advantages over the hard clustering approach:

1. It avoids the continuous pairwise distance computation by leveraging the progressive removal of already paired streams.

2. It eliminates almost completely the incorrect stream pairings.

The second goal is achieved by reducing the aggressiveness of the pairing protocol, which in practice will have a small impact on the convergence rate (compared to the hard clustering approach). Recall that the progressive algorithm classifies the stream with the maximum cimilarity value (max_1) as a match, if

$$\mathrm{max}_1 - \mathrm{max}_2 > f \cdot (\mathrm{max}_2 - \mathrm{cMass}).$$

The value f essentially tunes the algorithm's convergence rate. Smaller values of f mean that the algorithm is more elastic in its pairing decisions, hence achieving faster convergence, but possibly introducing a larger amount of incorrectly classified pairs. By imposing larger f values, we restrict the algorithm in taking more conservative decisions. This way fewer mistakes are made, at the expense of more prolonged convergence times.

In Figure 15 we present the accuracy of the Asymmetric-Jaccard using values of $f = 1/2, 2/3, 1$. The darker part of the graph indicates the correctly classified pairs, the medium gray the incorrect pairing, and the white part are the remaining streams for which no decision has yet been made. From the graph, one can observe that for the examined dataset $f = 2/3$ represents the best compromise between convergence rate and false pairing rate. The final pairing results after 300sec are: correctly paired $= 972$, incorrectly paired $= 6$, undecided $= 24$. Contrasting this with the hard clustering results at 300sec (correctly paired $= 982$, incorrectly paired $= 18$), we see that we can achieve fewer false assignments, while being quite competitive on the correct assignments and at the same time accomplishing a progressive clustering that is computationally less demanding.

7.3 Resilience to Latency

We conduct experiments which indicate that the matching quality is not compromised by potential end-to-end network delay. For simplicity of exposition we assume an end-to-end delay for each stream that remains constant as time passes (even though on a real network delay will vary over time).

For this experiment we assume that each stream experiences a different global latency, drawn randomly from a uniform distribution within the range $[0, 2\delta]$, where δ is the observed one-way network latency. We conduct 4 sets of experiments with values $\delta = 40, 80, 160, 240$msec, therefore the maximum possible synchronization gap between 2 pairing streams can be up to 2δ.

Figure 16 displays the pairing accuracy using the two clustering parameters that produce the least amount of misclassifications, $f = 2/3$ and $f = 1$. The 3D areas indicate the number of correctly paired streams, while on top of the surface we also indicate in parenthesis the number of incorrect pairings. We report the exact arithmetic values for the mid-point of the experiment (150sec) and at the end of the experiment (300sec). Generally, we observe that the clustering approach is robust even for large end-to-end latency. The accuracy of the pairing technique is not compromised, since the number of misclassified pairs does not increase. For latency of 40–80msec the correctly classified pairs still remain approximately around $970/1000$. This number drops slightly to $960/1000$ for 240msec of latency, but still the number of misclassified pairs does not change. Therefore, latency affects primarily the *convergence rate*, since ambiguity is increased, however accuracy is not compromised.

One can explain these results by noting that complementary similarity is most dominantly affected by the long speech and silence segments (and not by the very short ones). The long speech and non-speech patterns between conversing users are not radically misaligned by typical network end-to-end latencies, therefore the stream similarities in practice do not deviate significantly from their expected values.

Summarizing the experiments, we have shown that the progressive algorithm can achieve pairing accuracy that reaches 96–97%, while it can be tuned for faster conver-

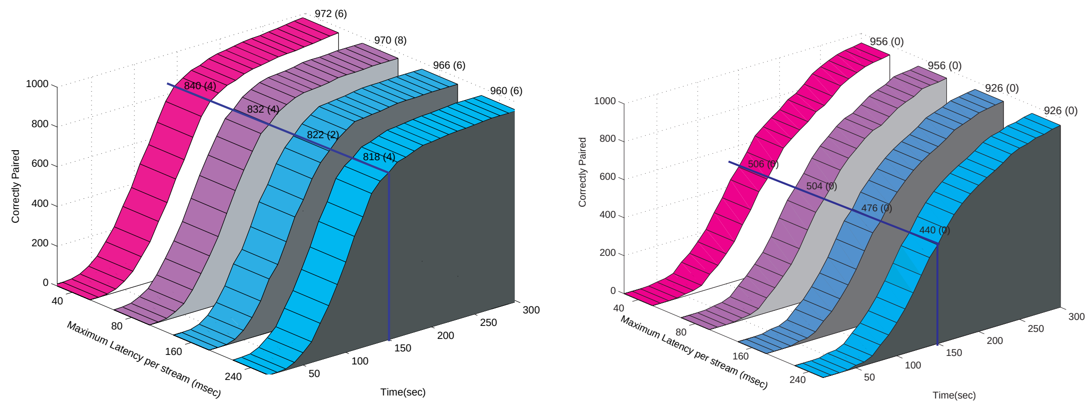

Figure 16: Accuracy of progressive pairing under conditions of end-to-end network delay. The graph depicts the correctly paired streams, while in the parenthesis we provide the number of incorrectly paired ones. Left: $f = 2/3$, Right: $f = 1$

gence or minimization of false classifications. More significantly, we have demonstrated that the clustering performance is not affected by the network latency, since latency does not significantly affect the dominant temporal dynamics between conversational patterns.

8 Conclusions

We have presented results indicating that intercepted VoIP data can potentially reveal private information such as pairs of conversing parties. Careful analysis of the voice packets coupled with an effective complementary pairing of voice activities can achieve high accuracy rates. We have also demonstrated that data encryption schemes cannot throttle the pairing of conversations. While we have attempted to examine the current problem from multiple aspects, many avenues are still open for investigation. Areas that that we are currently exploring are the provision for distributed execution of our pairing algorithm, as well as the fusion of multiple distance measures at different execution stages of the algorithm. The ultimate objective of such efforts are to provide a pairing algorithm that exhibits fast convergence, in addition to being robust and accurate. We believe that our algorithms and pairing models could be of independent interest, for general pairing of binary streaming data. Closing, we would like to point out that the main objective of this paper was not to suggest ways of intercepting VoIP traffic for malicious reasons, but merely to raise the awareness that privacy on Internet telephony can be easily compromised.

References

[1] S. Basu. *Conversational Scene Analysis*. PhD thesis, Massachusetts Institute of Technology, 2002.

[2] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The secure real-time transport protocol (srtp). In *IETF RFC 3711*, March 2004.

[3] A. Z. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences*, 1997.

[4] H. Clark and S. Brennan. Grounding in Communication. In *L. B. Resnick, J. Levine, & S. D. Teasley (Eds.), Perspectives on socially shared cognition*, pages 127–149, 1991.

[5] C. Clifton, M. Kantarcioglu, A. Doan, G. Schadow, J. Vaidya, A. K. Elmagarmid, and D. Suciu. Privacy-preserving data integration and sharing. In *DMKD*, pages 19–26, 2004.

[6] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing Data Streams Using Hamming Norms. In *IEEE Trans. Knowl. Data Eng. 15(3): 529-540*, 2003.

[7] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. In *Communications of the ACM*, volume 42, February 1999.

[8] D. Graff, K. Walker, and A. Canavan. Switchboard-2 phase ii. LDC 99S79 – http://www.ldc.upenn.edu/Catalog/, 1999.

[9] M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *SIGKDD*, pages 599–604, 2004.

[10] T. Li. A General Model for Clustering Binary Data. In *ACM SIGKDD*, 2005.

[11] C. Ordonez. Clustering Binary Data Streams with K-means. In *8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 12 – 19, 2003.

[12] R. V. Prasad, A. Sangwan, H. Jamadagni, C. M.C, R. Sah, and V. Gaurav. Comparison of voice activity detection algorithms for voip. In *Proc. of the 7th International Symposium on Computers and Communications (ISCC02)*, 2002.

[13] H. Sacks, E. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking in conversation. In *Language, 50*, pages 696–735, 1974.

[14] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer voip calls on the internet. In *ACM Conference on Computer and Communications Security (CCS)*, 2005.

[15] P. Zimmermann. Zfone – http://www.philzimmermann.com/zfone, March 2006.

[16] Findnot – `http://www.findnot.com`.