III: Small: Privacy-Preserving Techniques for Speech Processing
C. Project Description

C-1. Introduction to Privacy-Preserving Processing

The need to protect privacy needs no explanation: nearly every entity who possesses data desires to protect it from undesired inspection. For the most part, privacy concerns have related to protection of data from tapping or snooping during transmission or storage. Here both the sender (for storage this would be the person storing the data) and the intended recipient of the data are presumed to be allowed complete access to it, but it is necessary to protect it from being accessed by any unauthorized third party. An array of technologies have been developed to deal with this problem. Currently the most common methods are asymmetric public-key methods (Salomaa, 1996) that render the data into a form that is virtually indistinguishable from noise to the outside observer, and have been shown to require nearly impossibly long times to break, making communication between a sender and a receiver virtually impregnable in most circumstances.

A different type of situation arises when two or more parties desire to engage in some collaborative computation, but mistrust one another and do not wish to expose their data or any information they may gather to each other. A commonly-cited illustrative example of this is the so-called millionaire problem (Yao, 1982). Two millionaires desire to find out which of them is richer, but are unwilling to give each other any indication of what their actual net worth is. Many real-world examples of such problems are also readily found, such as distributed voting, where the goal is to obtain a consensus through voting over a network without revealing any individual vote, private bidding auctions where the outcome of the auction must be obtained without revealing any individual bid, private information retrieval where a person may desire to retrieve a specific information from a database without revealing the query to it, privacy-maintaining collaborative filtering where co-occurrence statistics must be obtained without having access to individual ratings etc. Such problems, where the desire to process data conflicts with need for privacy can theoretically be solved through ”fully Homomorphic” encryption schemes that enable computation of arbitrary functions directly on encrypted data (Rivest et al., 1978). However, despite recent developments (Gentry, 2009) computationally-feasible fully-homomorphic encryption remains elusive. Instead, one must employ *secure multi-party computation* (SMC) protocols (Lindell, 2003).

Secure multi-party computation was first introduced by Yao (1982) in his seminal paper in which he gave a solution to the millionaire problem. A multi-party protocol is one where a set of participants $p_1, p_2, \cdots, p_N$, possessing private data $d_1, d_2, \cdots, d_N$ respectively, wish to compute the value of an $N-$variable function F at $(d_1, d_2, \cdots, d_N)$. The protocol is deemed *secure* (i.e. an SMC) if no participant can learn more from the computation than what can be inferred from their own data and the final outcome of the computation. For instance, in the two-party millionaire problem, a person may only be able to determine if the other is more/less wealthy than themselves with no other indication to their actual worth. The general mechanism involves an iterated exchange of encrypted partial information between the parties according to a prespecified protocol until eventually one or more parties have the result. The protocols incorporate a variety of data-hiding methods such as homomorphic encryption, data randomization, oblivious transfer (Rabin, 1981), etc. In most scenarios zero-knowledge proofs (Quisquater et al., 1990) may also be instituted to ensure that all parties have obeyed the protocol.

SMC protocols have since been used to develop *privacy-preserving* computational methods for a variety of processing, learning and classification tasks, such as multiple parties performing k-means (Vaidya and Clifton, 2003), learning trends (Bahmani et al., 2010), building decision trees (Sutahmpan and Maneewongvatana, 2005), support vector machines (Zhan and Matwin, 2007) or other classifiers, or computation of means and related statistics from distributed databases (Kiltz et al., 2005), and rudimentary computer vision applications (Avidan and Butman, 2006), all without exposing any more information about the data than that which is available in the outcome of the processing. The literature on the topic is currently quite large – indeed, with due apologies to the authors, too large to describe here in any detail.

Although researchers have widely addressed the topic of preserving privacy in the context of processing data such as text, network traffic, images and numeric databases, little if any attention has been paid to preserving privacy during voice processing. Speech is regarded as of the most private forms of communication. People do not like to be eavesdropped on, or recorded without permission – in fact these are considered illegal in many situations. A person's voice contains information not just about what they spoke but also about their gender, nationality (accent) and even their emotional state, all of which they may not want to reveal. Yet, current voice processing systems that perform speaker verification or identification, speech recognition, or learning tasks require complete access to the speaker's voice. Not granting them access to one's voice is only possible if one does not use them at all.

C-2. Project Objectives: Privacy-Preserving Techniques for Voice

In this project we propose to develop privacy-preserving frameworks for processing *voice* data.

We first introduce some terminology that we will employ in the rest of this document. The term **voice processing** refers to pattern classification or learning tasks performed on voice data. A **privacy-preserving** transaction is one where no party learns anything about the other's data. In a voice processing system, this would mean that the system does not learn anything about the user's speech *and* the user does not learn anything of the system's internal parameters either. We often also refer to privacy-preserving operations as "secure". We expand the conventional definition of *system* to include the entity in charge of it (such as a corporation that operates a speaker verification system) who could log data and partial results for analysis.

Current voice processing technologies are not designed to preserve the privacy of the speaker. Systems need complete access to the voice, albeit in parameterized form. The only privacy ensured is the loss of information effected by the parametrization. Standard parametrizations can be inverted to obtain an intelligible speech signal and provide little protection of privacy.

Yet, there are many voice processing situations where there may be need to preserve the privacy of subjects' voices. Processing may need to be performed without having access to the voice. Here, "access to voice" refers to having access to any form of the speech that can be converted to an intelligible signal, or from which information about the talker or what they spoke could be inferred.

- **Speaker Verification:** Speaker verification systems determine if a speaker is indeed who he/she claims to be. Users may want the system not have access to their voice or identity. In text-dependent verification systems which use pass phrases, users may not want the system to be able to discover the actual passphrase.
- **Speaker ID:** Speaker identification systems attempt to identify which, if any, of a prespecified set of speakers has spoken into the system. In many situations, it would be useful to be permit speaker identification without providing access to any other information in the voice. For instance, a security agency may be able to detect if a particular speaker has spoken in a phone conversation without being able to discover who else spoke or what was spoken in the audio.
- **Keyword spotting:** Keyword spotting systems detect when a prespecified keyword has occurred in a recording. It may sometimes be useful to permit a keyword spotter to only detect if a particular keyword did occur, without granting access to any other information such as who spoke or what may have been spoken. This would be useful for privacy-preserving monitoring in surveillance applications, secure passphrase detection etc. It would also enable privacy preserving speech mining applications, which could search for and extract statistics about the occurrence of specific keyterms or phrases in an collection of recordings, but not learn anything else about the audio.
- **Speech Recognition:** It may sometimes be desired to recognize speech without actually observing it. One example is where users employ a remote recognition server and do not wish the server to have access to their voice.
- **Training and Enrollment:** Speech recognition systems require both the speech data and their transcriptions in order to be trained. A privacy-preserving training algorithm would not have true access to either the speech signal or its transcription. This would make it possible to train from large (possibly distributed) voice datasets that are considered private.

The list can be continued. The following are not strictly voice technologies such as those mentioned above, but are nevertheless voice-based applications that bring out the need for privacy-preserving processing.

- **Surveillance:** Surveillance systems monitor speech recordings in order to detect the presence of specific voices or the utterance of specific phrases. Privacy-preserving speaker ID or keyword/keyphrase spotting techniques would enable creation of an automated framework within which the monitoring agency would be notified when a specific voice or key phrase was detected (for which they may be searching with legal sanction), but never have access to any of the other information in the voice recordings. While this does not solve the problem that surveillance by its very nature is invasive, it would greatly mitigate fears of unauthorized detection of other patterns in innocent conversations.

- **Data Mining:** Corporations and call centers often have large quantities of voice data from which they may desire to detect patterns. However, privacy concerns prevent them from providing access to other "outside" companies that can mine the data for patterns. Privacy-preserving speaker ID and keyword/keyphrase spotting techniques (among others) can enable the outside company to detect patterns without learning anything else about the data.
- **Algorithm development on private data:** Governments and corporations currently legally possess large quantities of audio recordings that they desire to develop processing algorithms for; however they frequently lack the necessary expertise in house. Privacy concerns prevent them from contracting the research out since the data cannot be shared, necessitating long legal procedures of clearances and contracts. A privacy-preserving framework would enable outside researchers to work on the private data without actually having access to it.

Other such applications may be listed.

The objective of our proposal is to develop *privacy-preserving* techniques for some of the key voice processing technologies mentioned above. Specifically, we propose to develop privacy-preserving techniques for **speaker identification**, **speaker verification**, and **keyword spotting**. Using a combination of tools from cryptography and secure-multiparty computation (SMC) we will develop *secure* (*i.e.* privacy-preserving) procedures, so that the privacy of all parties (speakers, the system, and any other entity involved in the processing) is preserved.

We note that the thrust of this project is not the development of better voice-processing algorithms per-se. Our emphasis is on privacy. Our strategy will be to restructure the computations of *current* algorithms, embedding them within SMC protocols such that the operations are privacy preserving. Where necessary, we will develop (or choose) alternative algorithms or implementations that are inherently more amenable to having their computations restructured in this manner. The classification (or learning) performance of the secure (privacy-preserving) system itself will be no better than the original algorithms that are so restructured.

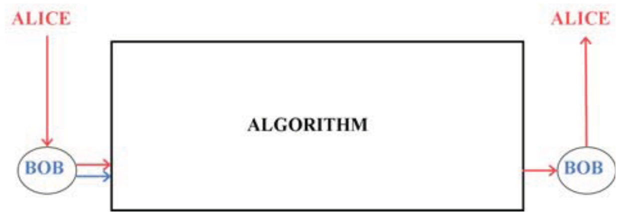
We believe that the proposed work is within the scope of a three-year project including a full-time student and part-time input and guidance from two senior investigators.

C-3. Technical Details

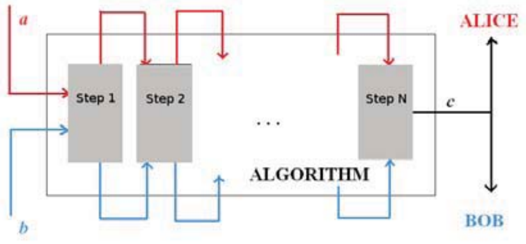
Our approach to privacy-preserving voice processing is based on secure multi-party computation (SMC) protocols. We restate the basic nature of SMC here for reference. In an SMC a set of participants $p_1, p_2, \cdots, p_N$, possessing private data $d_1, d_2, \cdots, d_N$ respectively, wish to compute the value of an $N-$variable function F at $(d_1, d_2, \cdots, d_N)$, such that no participant p_i can infer anything more about the data from participants $p_j, j \neq i$ than may be inferred from the final outcome of the computation. In the simplest instances the function $F()$ performs simple computations such as the primitives we describe in the next section.

C-3-1. Processing With Primitives

Any data processing algorithm proceeds through a sequence of steps. In voice-processing applications such as the ones described in Section C-2, the speaker typically hands off the speech signal to the system which monolithically performs all necessary steps and returns the result to the user. This is illustrated in Figure 1(a). The implicit assumption here is that the speaker trusts the system not to misuse their voice.



(a) Conventional implementation of voice processing system



(b) SMC implementation of voice processing system

Figure 1: (a) In a conventional implementation of a voice processing algorithm data from Alice the speaker (blue lines) are received by Bob the system. Bob processes Alice's data with his parameters (in red) and sends the outcome of the computation back to Alice (or a designated receiver). (b) In a privacy-preserving implementation, the algorithm is split into steps that are implemented as secure primitives (shown as gray boxes). Intermediate results are distributed as random additive shares and fed into the following steps. The final result is obtained by both parties (or the designated receiver).

In the privacy-preserving framework, the speaker (Alice) and the system (Bob) no longer trust one another. Alice is not interested in revealing her voice to Bob. Bob in turn has no desire to permit Alice access to his parameters (otherwise the problem would be rendered trivial; Bob could just ship his algorithm to Alice and her data remains secure with her). This may further be complicated through the addition of more parties to the computation. For instance, Charlie might be a third party who is the true intended recipient of the outcome of processing Alice's voice. Dave might be a participant who collaborates with Alice in training Bob's algorithm, but has no desire to let either of them access his voice.

In order to preserve privacy, Alice can no longer just ship her data off to Bob and expect him to perform the entire algorithm. Instead now the algorithm is performed in as a sequence of steps. Each of these steps is performed by an SMC protocol that distributes the results as "shares" between both parties such that neither can infer anything about the actual outcome from it. Figure 1(b) illustrates this schematically. Each of the individual steps accomplishes one of a small number of basic operations which we will refer to as primitives. We list some of these primitives below. Note that our arithmetic notation represents scalars as regular characters and vectors as bolded ones:

- **Secure Inner Products (SIP):** If Alice has a vector $\mathbf{x}$ and Bob has vector $\mathbf{y}$, the SIP protocol (SIP($\mathbf{x}, \mathbf{y}$)) produces two seemingly random numbers a and b such that $a + b = \mathbf{x}^\top \mathbf{y}$. Alice receives the share a and Bob receives b. These shares can either be propagated to subsequent steps of the algorithm directly or aggregated by a mutually acceptable party to obtain the final result. A variety of SIP algorithms have been proposed in the literature, *e.g.* Ioannidis et al. (2002) and Goethals et al. (2004).

- **Secure Max Index (SMAX):** Alice has a vector $\mathbf{x} = [x_1, x_2, \cdots]$ and Bob has $\mathbf{y} = [y_1, y_2, \cdots]$. They wish to compute the index of the maximum of $\mathbf{x} + \mathbf{y} = [x_1 + y_1, x_2 + y_2, \cdots]$. The SMAX protocol (Atallah et al., Oct 2003; Blake and Kolesnikov, 2004; Lin and Tzeng, 2005) delivers randomly permuted additive shares $\mathbf{a}$ and $\mathbf{b}$ of $\mathbf{x} + \mathbf{y}$ to Alice and Bob. The max index is obtained through an additional processing step that also ensures that the maximum value is not exposed. Alternately, the partial shares could be propagated as-is to subsequent steps of an algorithm.

- **Secure Max Value (SVAL):** Alice and Bob wish to compute the value of the largest component of $\mathbf{z} = \mathbf{x} + \mathbf{y}$. This can be achieved using the SVAL($\mathbf{x}, \mathbf{y}$) protocol (Atallah et al., Oct 2003), the outcome of which is to provide Alice and Bob additive shares of a and b of the maximum value, but no information about its index.

The primitives in turn have two essential steps: first both Alice and Bob encrypt their own data using special cryptographic techniques *e.g.* Goethals et al. (2004). The computation is performed on the encrypted data and results are distributed. When primitives are chained so that the output of one primitive is fed to the next, the computation can frequently be arranged so that the information passed from one step to the next can remain in the encrypted domain and repeated encryption is not necessary.
In addition to the above, it is typically useful to define some other *macro* operators *e.g.*

- **Secure Logsum (SLOG):** This is a rather unusual operation proposed by Smaragdis and Shashanka (2007a) as a useful macro for speech processing algorithms. Given that Alice and Bob hold vectors $\mathbf{x}$ and $\mathbf{y}$, the goal is to compute $z = \log \sum_i e^{x_i + y_i}$. The SLOG protocol once again provides additive shares of the result to Alice and Bob.

- **Secure Trend Computation (SPC):** This macro, proposed by Bahmani et al. (2010) permits Alice and Bob to get shares of principal components of the union of their data (or sufficient statistics) privately. It uses an arbitrator Trent and is an important tool for enrollment procedures.

Other similar operators must be developed as required by the algorithm.

C-3-2. Building Blocks

In (Shashanka and Smaragdis, 2006; Smaragdis and Shashanka, 2007a,b) we have described a series of secure algorithms for learning and classification with Gaussian Mixture Models and Hidden Markov Models. These form the basis of most current voice-processing algorithms. We describe these briefly below.

Gaussian mixture models

The logarithm of a Gaussian density with mean μ and covariance Σ defined over vectors of dimensionality D is given by

$$\begin{aligned} \log(G(\mathbf{x};\mu,\Sigma)) &= -0.5D\log(2\pi) - 0.5\log|\Sigma| - 0.5(\mathbf{x}-\mu)^\top\Sigma^{-1}(\mathbf{x}-\mu)) \\ &= C - 0.5\mathbf{x}^\top\Sigma^{-1}\mathbf{x} + \mathbf{x}^\top\Sigma^{-1}\mu - 0.5\mu^\top\Sigma^{-1}\mu) \end{aligned} \tag{1}$$

where C is a parameter that incorporates all constant terms. If Bob possesses the Gaussian and Alice has a vector $\mathbf{x}$, Shashanka and Smaragdis (2006) show how they can compute the value of the Gaussian at $\mathbf{x}$ securely, so that Alice never learns anything about μ and Σ while Bob learns nothing of $\mathbf{x}$. The key lies in noting that $-0.5\mathbf{x}^\top\Sigma_{k,i}^{-1}\mathbf{x}$ can be computed securely by Alice and Bob as $\mathbf{a},\mathbf{b} = SIP(\mathbf{x}, -0.5\Sigma^{-1})$; $d, e = SIP(\mathbf{x}, \mathbf{b})$; $f = d + \mathbf{a}^\top\mathbf{x}$, resulting additive shares f and e with Alice and Bob respectively. Similarly we get $g, h = SIP(\mathbf{x}, \Sigma^{-1}\mu)$. Eventually Alice and Bob end up with additive shares $i = f + g$ and $j = e + h + C$. i and j must be added to get the final value. Alice and Bob have not seen any of each other's data. We refer to this operation as $i, j = SecureGaussian(\mathbf{x};\mu,\Sigma)$.

A Gaussian mixture model is given by $GMM(\mathbf{x}) = \sum_i w_i G(\mathbf{x};\mu_i,\Sigma_i)$. The log of a Gaussian mixture can then be written as

$$\log GMM(\mathbf{x}) = \text{logsum}(\log w_1 + \log G(\mathbf{x};\mu_1,\Sigma_1)), \log w_2 + \log G(\mathbf{x};\mu_2,\Sigma_2), \cdots) \tag{2}$$

Shashanka and Smaragdis (2006) show how this can be securely computed. Alice and Bob obtain additive shares of $a_i, b_i = SecureGaussian(\log G(\mathbf{x};\mu_i,\Sigma_i))\forall i$, Bob adds $\log w_i$ to all his shares, and finally $q, r = SLOG([a_1, a_2, \cdots], [b_1 + \log w_1, b_2 + \log w_2, \cdots])$ provides Alice and Bob with shares q and r of the log GMM. We refer to this procedure as *SecureGMM*.

Classification with GMMs is now easily performed. Given a number of classes modelled by GMMs, Alice and Bob obtain additive shares q_i and r_i for the log GMMs of all classes using *SecureGMM*. Classification is performed as SMAX($[q_1, q_2, \cdots], [r_1 + \log P_1, r_2 + \log P_2, \cdots]$), where P_i is the *a priori* probability of the i^{th} class. The SMAX delivers shares of the result to Alice and Bob, and either they or a third party Charlie can obtain a final result. In the process no other information has been divulged to any party.

Smaragdis and Shashanka (2007a) also specify how a GMM could be trained similarly with data from multiple contributors such that no party sees any data besides their own and only Bob gets to know the parameters of the learned GMM.

Hidden Markov Models

Hidden Markov Models comprise a set of states $s_1, s_2, \cdots, s_N$, each with an associated output probability distribution function $P(\mathbf{x}|s_i)$. The parameters of the model are the set of initial state probabilities $\pi(s_i)$ which can be jointly represented as a vector Π, a set of transition probabilities which can be represented as a matrix A, and the parameters of the state output probabilities $P(\mathbf{x}|s_i)$. Typically the state output probability densities $P(\mathbf{x}|s_i)$ are a Gaussian mixture density.

Let $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$ represent the sequence of data vectors. Let $\mathcal{P}(\mathbf{x}) = [P(\mathbf{x}|s_1), P(\mathbf{x}|s_2), \cdots]$ be a vector formed from the output density values of individual states. It can be shown that the probability of $\mathbf{X}$ given the HMM can be computed as

$$P(\mathbf{X};\Theta) = I^\top(A^\top(\cdots(A^\top(A^\top(\Pi \otimes \mathcal{P}(\mathbf{x}_1)) \otimes \mathcal{P}(\mathbf{x}_2)) \otimes \mathcal{P}(\mathbf{x}_3))\cdots) \otimes \mathcal{P}(\mathbf{x}_T)) \tag{3}$$

where $\otimes$ represents the Hadamard component-wise multiplication, I represents a vector of ones and Θ represents the set of parameters for the HMM.

Smaragdis and Shashanka (2007b) show how the the above could be computed securely. If state output densities are GMMs, shares of $\log \mathcal{P}(\mathbf{x}_i)$ can be obtained using *SecureGMM*. Thereafter, Equation 3 simply represents a chain of SIP operations. Smaragdis and Shashanka (2007b) actually perform this operation in the log domain (for the usual reason of preventing underflows), so the operation becomes a sequence of SLOG operations. The outcome is that Bob (who holds HMM parameters) and Alice obtain additive shares of the final score.

The probability of the most-likely state sequence through the HMM, used in most HMM-based classification schemes, can similarly be computed by replacing some of the SLOG operations by the SVAL primitive (Smaragdis and Shashanka, 2007b). If the state sequence is also desired, the SMAX primitive is also required. Finally Alice and Bob receive shares of both the probability and the optimal state sequence.

As in the case of GMMs, classification is straightforward when individual classes are modelled by HMMs – Alice and Bob obtain shares of the scores of individual classes and the final outcome is obtained with the SMAX primitive. Training HMMs is slightly more involved than training GMMs. Equation 3 represents the computations of the *forward* algorithm for computing data probability using an HMM. The backward algorithm can be similarly defined. Smaragdis and Shashanka (2007a) show how this leads to a secure training algorithm that trains all parameters of the HMM from accumulated data from multiple consenting but privacy-desiring parties.

C-3-3. Work Plan

Voice-processing technologies including speaker identification, verification and keyword spotting typically follow a two-step process: signal parameterization followed by classification (both of which must be made privacy preserving).

The most common parametrizations for speech are mel-frequency cepstral coefficients (Davis and Mermelstein, 1980) and perceptual linear predictive (PLP) features (H.Hermansky, 1990). Often specialized features and transformations *e.g.* (Kumar and Andreou, 1998), as part of the classification mechanism.

The classification approaches for speaker verification are usually GMM or HMM based (Bimbot et al., 2004). In both approaches, the classifier compares a model of the distribution of the speaker's audio to a background model. Alternative classifiers such as SVMs (Zhao et al., 2007) have also been successfully used for speaker verification. The most effective current methods are "Gaussian supervector" techniques which combine the strengths of both GMM and SVM approaches (Campbell et al., 2006). Keyword spotting systems aim to detect specific spoken terms in a voice recording. Keyword spotting typically falls into two categories: a) the "garbage-model" approach, which compares a model for the desired keyword against an alternative background model (Chen et al., 2006), and b) the LVCSR approach based on large vocabulary continuous speech recognition. For this project, we will concentrate on variants of the former.

Not all algorithms are equally amenable to being cast in efficient privacy-preserving form. Hence it is advisable to also explore alternate methods that, while providing similar results to conventional techniques, are more amenable to efficient privacy-preserving implementation. E.g. the word-spotting technique based on spectro-temporal patterns proposed by Ezzat and Poggio (2008), performs at state-of-the-art levels using correlations and thresholds as primary mechanisms, both of which are well suited to privacy-preserving implementation. We plan to also explore alternative techniques such as these for privacy-preserving speaker verification and keyword spotting.

In the work plan we outline below, we present our ideas using one or more typical current techniques as illustrative exemplars; however it must be understood that investigation into the most *effective* classification scheme (which both provides accuracy *and* is amenable to efficient implementation in privacy-preserving form) is part of the proposed work in every case.

Parametrization

One of the key components of any voice processing system is the parametrization of the speech signal, by which feature vectors are derived from it. In a *privacy-preserving* system, the subject's voice must not be "heard" by the system. Since the feature vectors derived from the voice carry significant information (for instance, they could be used to recognize what the speaker said), they must also not be accessible to the system. A simple solution to this would be for the subject to perform the feature computation at their end (later processing maintains the privacy of these vectors); however frequently the parametrization too is a secret that the *system* may not desire to reveal. Consequently the parametrization must also be performed in a privacy preserving manner that preserves both the system's and the subject's privacy.

Most parametrization schemes can be expressed simply in terms of operations such as logarithms and inner products. For instance, MFCCs can be computed as $MFCC = D\log(M((FX) \otimes conjugate(FX)))$, where X represents a vector of signal samples, F is the Fourier transform in matrix form, M is the set of Mel filters represented as a matrix, and D is a DCT matrix. It is relatively simple to convert operations such as these to be expressed as a sequence of secure primitives. In this project we will assume that the parametrization derives MFCCs and develop a secure mechanism for computing MFCCs. These will be the features used, unless otherwise required. The outcome of the computation will be shares distributed between the speaker and the system, a feature that must be taken into account when securing the systems themselves.

Secure Speaker Verification

Speaker verification systems are biometric systems that attempt to verify that a person is indeed who they claim to be. The systems can be either text dependent, where the speaker is expected to utter a specific passphrase, or text independent, where anything may be said. A large number of classification algorithms such as those mentioned earlier in this section have been proposed in the literature for this task.

"Privacy" in this case can have two interpretations. In the first, the speaker may wish to be verified, but does not wish to allow the system to record his voice (or any features derived from it). In this case the simplest systems that can be converted to privacy-preserving form are GMM- and HMM-based ones since the solutions outlined in C-3-2 can be adapted for them, but solutions that use combinations of GMMs and SVMs (Campbell et al., 2006) may also be employed in conjunction with privacy-preserving SVM algorithms that we are currently developing. Various additional issues must also be considered. If the feature computation is secure, the system and the user possess additive shares of the data. The algorithms must be modified to account for these. Speaker verification schemes employ a variety of "anti-speaker" and "garbage" models. Scores computed from the various models contribute to normalization schemes that must also be incorporated securely (Bimbot et al., 2004). These normalization schemes sometimes include selection of the models themselves that will used by the system.

In the more second, more complex scenario, it may be desired that the user is correctly verified, but the system itself remains unaware of who exactly was verified. To do so, the privacy-preserving framework must keep *all* model types active at all times and the precise model used must not be revealed at any time. This introduces a complication since in addition to adding algorithmic complexity, it also adds computational complexity. If the situation warrants (and allows) for the actual identity of the user to also be hidden from the system, then the models for all, or at least some random subset of users must be always active to obfuscate the identity of the user.

Secure Speaker Identification

Speaker identification systems simply determine which if any of a pre-specified set of speakers a new recording belongs to. Speaker ID systems are very similar to speaker verification systems, with the primary difference being that while the latter is essentially a two-class classification problem, the former is a multi-class problem where each of the speakers in the set is a class. The techniques employed in the two cases are very similar – enrollment data from each of the speakers are used to build statistical or discriminative models for the speaker. These models are employed to recognize the class for a novel recording. A large variety of algorithms based on statistical models ranging from GMMs and HMMs to SVMs etc. have also been proposed for speaker identification (Reynolds, 2002; Hatch et al., 2006).

Given the similarity to speaker verification, the approach taken to the development of privacy-preserving speaker identification systems will also be similar. However, in this case we have an additional level of complexity. Like the speaker verification case, Alice's (who possesses the recording with the speaker's voice) data must be hidden from Bob (the system). In addition to that it may be required that the number of speakers in Bob's set must also be hidden, since Bob does not want Alice to know how many people he has models for – a difficult task since Alice participates in the computation. Also, the actual outcome of the identification may need to be handled differently – the intended recipient might be a third party, and not Alice or Bob. In order to deal with the first problem it may be required to separate the multiclass classification problem into a number of binary classification problems, e.g. as in Dietterich and Bakiri (1995). Alice remains unaware of the actual number of classes in the system. Since Bob must combine the output of all classifiers to determine the speaker's identity, and since these outcomes are shared with Alice, this provides a natural mechanism to enable only a third party (rather than Alice or Bob) to determine the outcome. An alternate solution may employ randomization mechanisms that can ensure minimum "leakage" of information to both Alice and Bob.

Secure Keyword Spotting

Keyword spotting systems attempt to detect if one or more of a specified set of key words or phrases has occurred in a recording. The common approach composes an HMM for each of the words, either directly from recordings of the word or from the components of a large-vocabulary recognizer. A generic "garbage" model representing all other speech is used in opposition to the model for the key words. Spotting is performed by comparing the likelihoods of the models for the key words to that of the garbage model on the speech recording *e.g.* Chen et al. (2006).

Keyword spotting systems that use word-level HMMs are analogous to HMM-based speaker identification systems in their operation, and the approach required to implement privacy-preserving versions of them will also be similar. However, when the system permits the user to specify the words and composes models for them from sub-word units an additional complexity is introduced if it is required that the identity of the keyword be hidden from the system. The approach will require composition of uninformative subword graphs and performing the computation in a manner that allows the recipient of the result to obtain scores from only the portions of the graph that represent the desired keyword. Secure graph-search algorithms (Brickell and Shmatikov, 2005) can be employed for this purpose.

A more fruitful approach may be that proposed by Ezzat and Poggio (2008). Here, the keyword spotting problem is reduced to computation of correlations and a simple SVM or boosted classifier. The approach is reportedly more accurate than the HMM-based system. Additionally, due to its simplicity, the complexity of privacy-preserving computation is also much lower for the method.

Secure Enrollment and Training

Speaker verification/identification systems need to be trained from enrollment data. Enrollment in verification/ID systems is frequently done by *adapting* a generic voice model (Reynolds, 2002). Adaptation could be transform-based or based on maximum *a posteriori* adaptation. Fortunately, most of these operations can be expressed as a sequence of operations such as inner products, Gaussian computations etc. and it can be expected that the algorithms can be converted to secure form relatively simply.

C-3-4. Practical Issues

Efficiency: Privacy comes at a cost. Privacy-preserving algorithms tend to carry large computational and communication overhead from encryption and decryption, and repeated communication of partial results, which tend to dominate the computational time requirements. We refer the reader back to Figure 1(b). The greater the number of blocks in the figure, the finer the granularity of the operation and the greater the control over privacy. On the other hand, each additional block is likely to introduce an additional pass of encryption and decryption, in addition to one or more rounds of communication. In order to reduce this overhead, it becomes necessary to take various steps that employ protocols with innately lower overhead, e.g. algebraic (Ioannidis et al., 2002) instead of cryptographic (Goethals et al., 2004) protocols for primitives such as SIP, devising procedures that can use symmetric encryption schemes rather than asymmetric ones, replacing encryption with other schemes such as randomization and oblivious transfer, etc. In addition, it becomes necessary to factor the algorithm itself to have fewer blocks (Figure 1(b)). All of these steps innately reduce the security of the algorithms. This tension between privacy and efficiency is a common theme of secure communication or computation. In the project we propose to err towards the side of efficiency, since a highly secure algorithm that is impractical is useless. However, in future work (or, if time permits, even within this project), we plan to continue to investigate how our algorithms can be made more secure, given improved computation and communication and improvements in SMC and homomorphic encyrption techniques.

Privacy: In most of our work, we will assume the participants to be honest, but curious. They will follow the prescribed protocol, although they may store and analyze the partial results they obtain. A *malicious* participant or imposter may attempt to subvert the computations or probe other participants in an attempt to access their data. While there are methods that can detect or protect against such intrusion *e.g.* Quisquater et al. (1990) they significantly decrease the efficiency of the protocols. We will not address this issue in our work.

The processes will be designed to be only *semantically secure*. For the voice-based applications we investigate we do not believe that this is likely to be a significant issue in the short term, although it may become important in the longer term.

Accuracy: Algorithms can (and will probably) be made more secure by *approximating* various intermediate steps to have more efficient privacy-preserving implementations. Efficiency concerns will also govern the choice of the basic algorithm that will be converted to privacy-preserving form – for instance we may prefer to use simpler algorithms that are better suited for privacy-preserving implementations. This will naturally result in lowered performance accuracy. This tradeoff between accuracy and efficiency (and privacy) will be investigated as part of this project.

C-4. Evaluation

The developed algorithms will be evaluated for correctness and efficiency using standardized databases. Correctness will be evaluated by comparing the output of secure implementations to conventional (ideal) implementations. Efficiency will be measured in terms of the increase in computation time as a result of introducing security. Efficient implementations are however not a primary goal in our work. The *security* of privacy-preserving systems is difficult to estimate since rigorous practical frameworks for such evaluation are a project unto themselves and outside the scope of the current project. Rigorous evaluation will primarily come through peer reviews of referred articles and software that we will publish.

C-5. Timelines and Milestones

The proposed duration of the project is 3 years. Per year, the project will involve the equivalent of 2 months of full-time work by two senior investigators, and year-round full-time work (including the academic year and summer) by a graduate student.

The tasks include development of baselines (with unsecured algorithms), selection of the best algorithm for conversion to privacy-preserving form, and development of privacy-preserving versions of these algorithms. The latter will include development of necessary primitives and macros. The tasks will also include efficiency analysis, designing for efficiency where possible, and actual implementations. The broad yearly milestones are planned as given below. The timelines are based on the senior investigators' prior experience with research on secure classification and are believed to be realistic.

First Year:

- Secure algorithms for feature computations. This will also familiarize the student with concepts of secure processing. (3 months)
- Preliminary development of a baseline speaker verification and identification systems. Performance must be close to, but need not be equal to state-of-art. The training/enrollment is not expected to be secure at this time. (3 months)
- Investigation of algorithms for secure implementation of speaker verification. (3 months)
- Algorithm development for speaker verification while securing subject identity. (3 months)
- Efficiency analysis and implementation. (concurrent with algorithm development)

Second Year:

- Privacy-preseving speaker verification. (Continues from year 1, 3 months)
- Analysis of algorithms for speaker recognition to identify most promising candidates. (2 months)
- Basic two-party secure speaker recognition system. (4 months)
- Extensions to hide the number of speakers in the system's list. (3 months)
- Investigation into issues relating to efficiency. (Extends through the year)

Third Year:

- Baseline implementation of (insecure) keyword spotting system. Performance will be close to, but need not be at par with state of art. (2 months)
- Basic implementation of secure keyword spotter with fixed set of keywords. (7 months)
- Secure enrollment into speaker verification and identification systems (3 months)
- Security vs. Efficiency issues. (Extends through the year)

C-6. Dissemination

The algorithms developed will be presented in refereed conferences and journals throughout the course of the project. The PIs will maintain a code repository with archival documentation that will be made available to researchers over the web. We also expect to develop coursework centered on the topic of the project, and the material for the coursework will be made public if permitted.

C-7. Intellectual Merit

This project addresses a hitherto unvisited problem – the privacy of the subjects who use voice-based systems such as speaker verification or recognition systems, speech recognition systems etc. The proposed research initiates an entirely new direction in the field of speech processing that draws on the fields of speech recognition, computer security and secure multiparty computation.

C-8. Broader Impact

The proposed technology has broad implications not just to speech science, but to society at large. As voice technologies proliferate, people have increasing reason to distrust them. With increasing use of speech-recognition-based user interfaces, speaker-verification-based authentication systems, and automated systems for everything from routing calls to purchasing airline tickets, the ability of malevolent entities to capture and misuse a person's voice has never been greater, and this threat is only expected to increase. The fallout of such misuse can be far greater than mere loss of privacy and have severe economic and social impact as well. This project represents a proactive effort at developing technologies to secure voice processing systems to prevent such misuse.

The project has the potential to seriously impact national security. The following headline appeared on ABC news in October 2008: "Exclusive: Inside Account of U.S. Eavesdropping on Americans". The article goes on to say: "*Despite pledges by President George W. Bush and American intelligence officials to the contrary, hundreds of US citizens overseas have been eavesdropped on as they called friends and family back home...*". This highlights a serious problem: the NSA desires to monitor telephonic conversations to determine if any of them are of import to national security, but this has resulted in an intolerable invasion of the privacy of hundreds of regular citizens. Presumably, the security agency is mainly interested in detecting certain voices, or identifying certain key phrases. However, even if they did so using automated systems, current technologies would still provide them full access to the subjects' recordings. Based on the methods proposed in this project, we envision a mechanism whereby they could obtain publicly-accepted forms of legal sanction to look for prespecified voices or phrases. The privacy-preserving framework ensures that they are only notified when these occur, but will have no access to the voice data itself, thus preserving citizens' privacy. While this is not a perfect solution since it will depend on the accuracy of the actual speaker ID or keyword spotting algorithm that has been secured, and the chosen keywords may also occur in innocent conversation, it is still a vast improvement over the current situation.

This work also has implications for voice research. Governments and corporations currently legally possess large quantities of audio recordings that they desire to develop processing algorithms for, but lack the necessary expertise in house. Privacy concerns prevent them from contracting the research out since the data cannot be shared, necessitating long legal procedures of clearances. Our work will enable them to permit outside researchers to work on these data without actually having access to it. Many other areas of broad impact can similarly be identified.

C-8. Prior NSF support

The investigators for this project all come from industry background where they were not required to solicit external funding. Consequently they have no prior NSF support. However they all have significant experience in conducting research, directing research teams including students and delivering completed projects as explained in supplementary documentation.