PRIVACY-PRESERVING SPEAKER VERIFICATION AS PASSWORD MATCHING

Manas A. Pathak and Bhiksha Raj

Carnegie Mellon University, Pittsburgh, PA, USA

{manasp,bhiksha}@cs.cmu.edu

ABSTRACT

We present a text-independent privacy-preserving speaker verification system that functions similar to conventional password-based authentication. Our privacy constraints require that the system does not observe the speech input provided by the user, as this can be used by an adversary to impersonate the user in the same system or elsewhere.

We represent the speech input using supervectors and apply locality sensitive hashing (LSH) to transform these into bit strings, where two supervectors, and therefore inputs, are likely to be similar if they map to the same string. This transformation, therefore, reduces the problem of identifying nearest neighbors to string comparison. The users then apply a cryptographic hash function to the strings obtained from their enrollment and verification data, thereby obfuscating it from the server, who can only check if two hashed strings match without being able to reconstruct their content. We present execution time and accuracy experiments with the system on the YOHO dataset, and observe that the system achieves acceptable accuracy with minimal computational overhead needed to satisfy the privacy constraints.

Index Terms— Privacy, Speaker Verification, LSH

1. INTRODUCTION

Speech being a unique characteristic of an individual is widely used as the biometric of choice in authentication systems. In this paper we investigate a text-independent speaker verification system that can authenticate while maintaining *privacy* over the speech data.

We consider a client-server model, where the speaker verification system is the server, and the user executes a client program on a network-enabled computation device such as a computer or a smartphone. Speaker verification proceeds in two phases: enrollment, where the user submits a few speech samples to the system, and verification where the user submits a test sample and the system makes a accept/reject decision based on its similarity to the enrollment data for that user. Our primary privacy constraint is that the verification system should not be able to observe the speech samples provided by the user both during the enrollment and verification steps. This is important as the same speech samples can potentially be used to verify the user in another authentication system. A speaker verification system might be compromised by an adversary that gains access speech samples from the internal storage of the system for this purpose. Similarly, the system itself might be made to pose as a front to phish speech data from the unaware users. To prevent this, we require that the speech samples should be transformed by the user via cryptographic operations so that they are irreversibly obfuscated from the system.

This work was supported by the NSF grant 1017256.

Our secondary privacy constraint is related to the computation device that is employed to the user in the verification process. We assume the user to be honest during the enrollment step, as it is in the user's privacy interest to be so. In the verification step, however, the user's computation device may be stolen or compromised by an adversary and can be used to impersonate as the user. To prevent this, we require that no data that may allow the adversary to authenticate itself in place of the user should be stored on the device. What we do not consider as a privacy violation is an attempt to directly trick an authentication system. An imposter might try to imitate a user's voice or use a recording to produce speech similar to the user. Although these attacks on speaker verification system do pose a critical threat, we consider them as a *security* issue, as the privacy of the original user's speech is not compromised in the ways mentioned above. Similarly, a verification system may arbitrarily deny a legitimate user despite having submitted a reasonably authentic sample of her voice. Again, for the same reasons we do not consider this as a privacy violation.

Pathak and Raj [1] introduce the problem of privacy preserving speaker verification and presented a solution using adapted GMMs. In this work, secure multiparty computation (SMC) protocols for adaptation and verification were developed using homomorphic encryption to enforce the privacy requirements. The main drawback of this approach was the huge computational overhead as compared with the non-private speaker verification system, which can make it impractical to be used in a realistic setting. This computation overhead is dependent on the length of the speech sample, requiring multiple minutes to hours for processing a few seconds of speech.

In this paper we propose a method for speaker verification that will require a minimal computation overhead needed to satisfy the privacy constraints. The central aspect of our approach is to reduce the speaker verification task to string comparison. Instead of using the UBM-GMM approach, we convert the utterances into supervector features [2] that are invariant with the length of the utterance. By applying the locality sensitive hashing (LSH) transformation [3] to the supervectors, we reduce the problem of nearest-neighbor classification into string comparison. It is very efficient to perform string comparison with privacy, similar to a conventional password system. By applying a cryptographic hash function (*e.g.*, SHA-256), we convert the LSH transformation to an obfuscated string which the server cannot use to gain information about the supervectors, but is still able to compare if two strings are identical. This one-way transformation preserves the privacy of the speech utterances submitted by the user, and is significantly more efficient to execute as compared to applying homomorphic encryption.

We emphasize that the main goal of this paper is not to develop a speaker verification algorithm that achieve higher accuracy. We are principally interested in developing an efficient speaker verification system that satisfies the same privacy constraints as [1] but with a minimal computational overhead and while achieving feasible

accuracy. Using the LSH functions defined over Euclidean and cosine distances, we show that our system achieves an equal error rate (EER) of 11.86% on the YOHO dataset, while requiring only a few milliseconds of computational overhead. We, nevertheless, consider this result as a proof of concept; we believe the EER can be further improved by using more discriminative speech features, better quality training data, and supervectors computed over a larger number of Gaussian components, all while utilizing the same private string comparison framework. Although UBM-GMM and SVMs trained over supervectors are known to supersede our current accuracy, our proposed algorithm significantly supersedes the computational overhead of the privacy-preserving variants of these algorithms.

The reminder of the paper is organized as follows. We present an overview of the supervector construction and locality sensitive hashing in Section 2. We discuss our privacy model in detail along with the privacy preserving enrollment and verification protocols in Section 3. We present the results of experiments with the protocols for accuracy and execution time on the YOHO dataset in Section 4.

2. PRELIMINARIES

2.1. Speaker Verification using Supervectors

We briefly summarize the supervector method for speaker verification introduced by Campbell et al. [2].

Adapted Gaussian mixture models (UBM-GMM) [4] is a well established method for performing speaker verification. A universal background model (UBM) is trained on a large and diverse set of speech samples, and is adapted using enrollment data from individual speakers by maximum *a posteriori* estimation to obtain the speaker models. A given test utterance is evaluated over both the UBM and the speaker model and the verification decision is made accordingly.

Campbell et al. [2] extend this approach by constructing a *supervector* (SV) for each utterance by performing the MAP adaptation of the UBM over that utterance and concatenating the means of the adapted model. Given an adapted model λ with M-mixture components of the form

$$P(x_t|\lambda) = \sum_{j=1}^{M} w_j \mathcal{N}\left(x_t; \mu_j, \Sigma_j\right),$$

where x_t is a frame in the utterance x, μ_j and Σ_j are mean vector and covariance matrices of the j^{th} component respectively, the supervector s is given by $s = (\mu_1 \parallel \mu_2 \parallel \cdots \parallel \mu_M)$.

This supervector is then used as a feature vector instead of the original utterance. The verification is performed using an binary support vector machine (SVM) classifier for each user trained on supervectors obtained from enrollment utterances and impostor data as the opposite class. As the classes are usually not separable in the original space, [2] also provide a linear kernel approximating KL-divergence between the two GMMs that is shown to achieve higher accuracy.

Instead of using SVMs with kernels, in this paper we use k-nearest neighbors trained on supervectors as our classification algorithm. The reasons for this choice are that, firstly, k-nearest neighbors also perform classification with non-linear decision boundaries and are shown to achieve accuracy comparable to SVMs with kernels [5]. Secondly, by using the LSH transformations we discuss below, it is possible compute approximate k-nearest neighbors by performing string comparison.

2.2. Locality Sensitive Hashing

Locality sensitive hashing (LSH) [3, 6] is a commonly used technique for performing efficient approximate nearest-neighbor search. An LSH function $L[\cdot]$ proceeds by applying a random transformation to a data vector x by projecting it to a vector $L[x]$ in a lower dimensional space, which we refer to as the LSH *key* or *bucket*. A set of data points that map to the same key are considered as approximate nearest neighbors.

As a single LSH function does not group the data points into fine-grained clusters, we use a hash key obtained by concatenating the output of k LSH functions. This k-bit LSH function $L[x] = L_1(x) \cdots L_k(x)$ maps an d-dimensional vector into a k-length string. Additionally, we use l different LSH keys that are computed over the same input to achieve better recall. Two data vectors x and y are said to be neighbors if at least one of their l keys, each of length k, matches exactly.

A family of LSH functions is defined for a particular distance metric. A hash function from this family has the property that data points, that are close to each other as defined by the distance metric, are mapped to the same key with high probability. Although there exist LSH constructions for a variety of distance metrics, including arbitrary kernels [7], in this paper we consider LSH for Euclidean distance (E2LSH) [8] and cosine distance [9] as they are simple to implement and are comparatively more efficient. The LSH construction for Euclidean distance with k random vectors transforms a d-dimensional vector into a vector of k integers, each of which is a number between 0–255. The LSH construction for cosine distance with k random vectors similarly transforms the given data vector into a binary string of length k.

3. PRIVACY PRESERVING SPEAKER VERIFICATION

3.1. System Architecture & Privacy Model

The speaker verification proceeds in two distinct phases: enrollment, where each user submits the speech data to the system, and verification, where a user submits a test utterance to the system which computes a yes/no decision as output. As discussed in Section 2.1, the users convert their speech utterances into supervectors and then apply l LSH functions. To facilitate this, the system provides with a UBM trained on publicly available data along with the random feature vectors constituting the LSH functions. There is no loss of privacy in publishing these elements, as none of these depend on the speech data belonging to the user.

Our first privacy constraint requires that the system should not be able to observe both the enrollment data and the test input provided by the user. Although LSH can be considered as dimensionality reduction, it is inherently non-privacy preserving. Due to the locality sensitive property of LSH, it is possible to reconstruct the input vector by observing a sufficient number of LSH keys obtained from the same vector. We satisfy this privacy constraint by requiring the users to apply a cryptographic hash function $H[\cdot]$ to the LSH computed over the supervector s, which we denote by $H[L(s)]$. Cryptographic hash functions satisfy the property that they can be computed efficiently, *i.e.*, in polynomial time on all inputs, but are computationally hard to invert, *i.e.*, there is no feasible algorithm to obtain the input $L(s)$ for a given output $H[L(s)]$. Cryptographic hash functions, such as SHA-256, MD5, are orders of magnitude faster to compute as compared to homomorphic encryption, such as the Paillier cryptosystem which is used in [1].

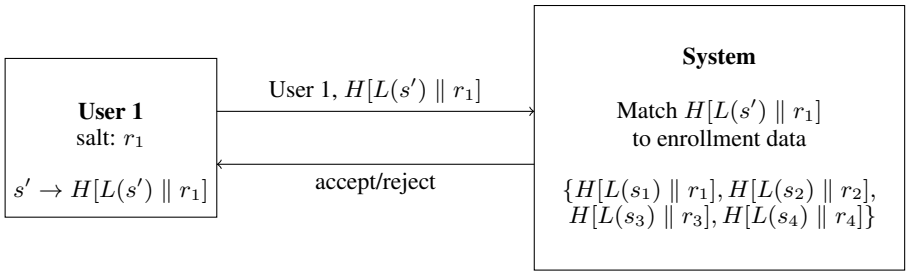

Fig. 1. System Architecture. For user 1, test utterance supervector: s', salt: r_1. Although only one instance of LSH function L is shown, in practice we use l different instances.

Salted Cryptographic Hash

As the possible values for LSH keys lie in a relatively small set by cryptographic standards, 256^k for k-bit Euclidean LSH and 2^k for k-bit cosine LSH, it is possible for the server to obtain $L(s)$ from $H[L(s)]$ by applying brute-force search. To make this attack infeasible, we vastly increase the domain of the hash function $H[\cdot]$ by concatenating the LSH key with a long random string r_i unique to the user i, which we refer to as the *salt*.[1] By requiring the user to keep the salt private and unique to each system, this also gives us the additional advantage of the cryptographically hashed enrollment data being rendered useless to an adversary. An adversary who has gained access to the system data will be unable to use the hashed enrollment data while trying to impersonate the user in another system. It is important to note that applying cryptographic hash function with salting has no effect on verification accuracy. If two LSH keys $L(s_1)$ and $L(s_2)$ match, their respective hashes salted with r: $H[L(s_1) \parallel r]$ and $H[L(s_2) \parallel r]$, would also match exactly, and vice-versa.

Our secondary privacy constraint is that an adversary should not be able to impersonate the user by having access to a compromised or stolen computation device belonging to that user. To satisfy this, we require the device to have no record of the enrollment samples and the previously used test samples. Although the salt r_i is stored on the device, it does not result in any loss of privacy by itself. The only point of failure in the system is when both the user device and the server data is compromised by the same adversary, who can then use the salt and the cryptographically hashed enrollment data to obtain the original LSH keys via brute-force search.

3.2. Protocols

We present the details of the enrollment and verification protocols below.

A. **Enrollment Protocol**

Each user is assumed to have a set of enrollment utterances $\{x_1, \ldots, x_n\}$. The users also obtain the UBM and the l LSH functions $\{L_1(\cdot), \ldots, L_l(\cdot)\}$, each of length k-bit from the system. Each user i generates the random 80-bit salt string r_i.

For each enrollment utterance x_j, user i:

(a) performs adaptation of x_j with the UBM to obtain supervector s_j.

(b) applies the l LSH functions to s_j to obtain the keys $\{L_1(s_j), \ldots, L_l(s_j)\}$.

(c) applies the cryptographic hash function salted with r_i to each of these keys to obtain $\{H[L_1(s_j) \parallel r_1], \ldots, H[L_l(s_j) \parallel r_1]\}$, and sends them to the server.

B. **Verification Protocol**

For a test utterance x', user i:

i. performs adaptation of x' with the UBM to obtain supervector s'.

ii. applies the l LSH functions to s' to obtain the keys $\{L_1(s'), \ldots, L_l(s')\}$.

iii. applies the cryptographic hash function salted with r_i to each of these keys to obtain $\{H[L_1(s') \parallel r_1], \ldots, H[L_l(s') \parallel r_1]\}$, and sends it to the server.

iv. The server compares the hashed keys for the test utterance with the hashed keys of the enrollment utterances, and counts the number of matches. Depending on whether this number is above or below a threshold, the server makes an accept/reject decision.

As discussed in Section 2.2, we consider two vectors to be matched if any one of their LSH key matches. The server calibrates the acceptance threshold by experimenting with match counts on held-out data.

While the above enrollment and verification protocols only consider enrollment data belonging to the speaker, it is also possible to include imposter data in the enrollment step. The user can similarly obtain supervectors from publicly available imposter data, apply LSH and the salted cryptographic hash function, and submit the hashed keys to the server. In the verification protocol, the server can match the test input separately to the hashed keys belonging to both the user enrollment and imposter sets, and make the decision by comparing the two scores. In Section 4, we observe that there is an improvement in performance by including the imposter data.

The server does not observe any LSH key before a salted cryptographic hash function is applied to it. Apart from the salt, the user does not need to store any speech data on its device. The enrollment and verification protocols, therefore, satisfy the privacy constraints discussed above.

4. EXPERIMENTS

We experimentally evaluate the privacy preserving speaker verification protocols described above for accuracy and execution time. We perform the experiments on the YOHO dataset [10].

[1] In practice, random strings of size 80-bits are used as salts.

4.1. Accuracy

We use Mel-frequency cepstral coefficient (MFCC) features augmented by differences and double differences, *i.e.*, representing a recording x by a sequence of 39-dimensional feature vectors $x_1, \ldots, x_T$. Although in practice the UBM is supposed to be trained on publicly available data, for simulation, we trained a UBM with 64 Gaussian mixture components on a random subset of the enrollment data belonging to all users. We obtained the supervectors by individually adapting all enrollment and verification utterances to the UBM.

Table 1. Average EER for the two enrollment data configurations and three LSH strategies: Euclidean, cosine, and Combined (Euclidean & cosine).

Enrollment: Only Speaker

Euclidean	Cosine	Combined
15.18%	17.35%	13.80

Enrollment: Speaker & Imposter

Euclidean	Cosine	Combined
15.16%	18.79%	11.86%

We use *equal error rate* (EER) as the evaluation metric. We observed that the lowest EER was achieved by using $l = 200$ instances of LSH functions each of length $k = 20$ for both Euclidean and cosine distances. A test utterance considered to match an enrollment utterance if at least one of their LSH keys matches. The score for a test utterance is given by the number of enrollment utterances it matched. We report the average EER for different speakers in Table 1. We observe that LSH for Euclidean distance performs better than LSH for cosine distance. We also used combined LSH scores for Euclidean and cosine distances and found that this strategy performed the best. This can be attributed to the fact that different distance measures find approximate nearest neighbors in different parts of the feature space. We hypothesize that the EER can be further improved by combining LSH functions defined over an ensemble of distance metrics. We leave this direction for future work.

We also consider two configurations where only the enrollment data of the speaker was used, and where imposter data chosen randomly from the enrollment data of other speakers was used along with the enrollment data of the speaker. In the latter experiment, we compare the difference between the number of utterances matched by the test utterance in the enrollment and the imposter set. We observed that using imposter data achieved lower EER when using the combined scores for both the distances.

Although it is known that conventional speaker verification approaches like UBM-GMM achieve higher accuracy, we consider the current performance of our system (EER 11.86%) as indicative of the feasibility of our approach. We believe this accuracy can be significantly improved by using more discriminative speech features, UBM trained over better quality training data and 1024 mixture components, and the combining LSH for multiple distance measures as discussed above. All these modifications can be directly included in the current framework while still having a minimal computational overhead for privacy.

4.2. Execution Time

As compared to a non-private variant of a speaker recognition system based on supervectors, the only computational overhead is in applying the LSH and salted cryptographic hash function. For a $64 \times 39 = 2496$-dimensional supervector representing a single utterance, the computation for both Euclidean and cosine LSH involves a multiplication with a random matrix of size 20×2496 which requires a fraction of a millisecond. Performing this operation 200 times required 15.8 milliseconds on average.[2]

The Euclidean and cosine LSH keys of length $k = 20$ require 8×20 bits = 20 bytes and 20 bits = 1.6 bytes for storage respectively. Using our C++ implementation of SHA-256 cryptographic hashing algorithm based on the OpenSSL libraries [11], hashing 200 instances of each of these keys in total required 28.34 milliseconds on average. Beyond this, the verification protocol only consists of matching the 256-bit long cryptographically hashed keys derived from the test utterance to those obtained from the enrollment data.

In this way, the enrollment and verification protocols add a very small overhead to the non-private computation. This is significantly smaller than the overhead of secure multiparty computation approaches using homomorphic encryption such as the UBM-GMM verification system of Pathak and Raj [1], while satisfying the same privacy constraints.

5. REFERENCES

[1] Manas Pathak and Bhiksha Raj, "Privacy preserving speaker verification using adapted GMMs," in *Interspeech*, 2011.

[2] William M. Campbell, Douglas E. Sturim, Douglas A. Reynolds, and Alex Solomonoff, "SVM based speaker verification using a GMM supervector kernel and NAP variability compensation," in *ICASSP*, 2006.

[3] Piotr Indyk and Rajeev Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *ACM Symposium on Theory of Computing*, 1998, pp. 604–613.

[4] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[5] Johnny Mariéthoz, Samy Bengio, and Yves Grandvalet, *Kernel Based Text-Independent Speaker Verification*, John Wiley & Sons, 2008.

[6] Alexandr Andoni and Piotr Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, pp. 117–122, 2008.

[7] Brian Kulis and Kristen Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *ICCV*, 2009, pp. 2130–2137.

[8] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *ACM Symposium on Computational Geometry*, 2004, pp. 253–262.

[9] Moses Charikar, "Similarity estimation techniques from rounding algorithms," in *ACM Symposium on Theory of Computing*, 2002.

[10] Joseph P. Campbell, "Testing with the YOHO CD-ROM voice verification corpus," in *ICASSP*, 1995, pp. 341–344.

[11] "OpenSSL," `http://www.openssl.org/docs/crypto/bn.html`.

[2] We performed all the execution time experiments on a laptop running 64-bit Ubuntu 11.04 with 2 GHz Intel Core 2 Duo processor and 3 GB RAM.