PRIVACY-PRESERVING MUSICAL DATABASE MATCHING

*Madhusudana Shashanka**

Boston University Hearing Research Center
677 Beacon St, Boston, MA 02215
shashanka@cns.bu.edu

Paris Smaragdis

Mitsubishi Electric Research Laboratories
201 Broadway, Cambridge, MA 02139
paris@merl.com

ABSTRACT

In this paper we present an illustratory process which allows privacy-preserving transactions in the context of musical databases. In particular we address the problem of matching a piece of music audio to a service database in such a way such that the database provider will not directly observe the query, nor its result, thereby preserving the privacy of the inquirer. We formulate this process within the field of secure multiparty computation and show how such a transaction can be achieved once we derive secure versions of basic signal processing operations.

1. INTRODUCTION

With the advent of digital music formats stored in networked computers we have seen the birth of an extensive industry offering online services. As networking efficiency increases we will undoubtedly see more and more musical services offered as online transactions. However this model suffers from the fundamental limitation that data needs to be exchanged in the clear thereby raising privacy concerns.

In this paper we examine this problem and present a framework within which one can allow other parties to perform signal processing operations on private data without raising any privacy concerns. We will demonstrate this principle using a simplified music database query example. We will show how to lookup a song title of an audio stream from another party's database without having to share the stream and without allowing the database owner to see the result of the matching.

Although such a constraint seems unrealistic we will show how it can be satisfied using Secure Multiparty Computation (SMC) protocols. These protocols arose from the cryptography community and allow multiple parties to perform arbitrary collaborative computations while guaranteeing the privacy of their data. This field originated from the millionaire problem where two millionaires want to compare their fortunes but are reluctant to disclose specific numbers to each other. A solution to this problem was first proposed by Yao [1] in the early 80s and since then we have seen the development of multiple variants of this idea in escalating complexity. The machine learning community has embraced this concept and employed it for various tasks such as multiple parties performing k-means [2], computation of means and related statistics from distributed databases [3] and rudimentary computer vision applications [4]. See [5] for a detailed treatment of the topic.

In this paper we present the application of SMC principles on the simple signal processing task facilitating song matching. We show how this can be a viable approach for a secure collaborative signal processing and how it can benefit online transactions where privacy is required.

The rest of the paper is organised as follows. In section 2 we introduce the problem at hand, the desired form of the solution, and the privacy constraints that need to be satisfied. In section 3 we provide a description of the field of SMC and the fundamental principles that we employ, and in section 4 we show the exact process used to solve the problem we pose in section 2. Finally we conclude and offer some pointers on how this particular example can be used for other kinds of privacy-preserving transactions involving audio processing.

2. PROBLEM FORMULATION

Consider two parties Alice and Bob. Alice has a CD recording but no information regarding the song title, artist, album name etc. Bob on the other hand has an extensive database of CDs which includes audio data alongside the kind of information that Alice seeks. Ultimately Alice wants to compare her audio file to Bob's collection and obtain the track information she seeks.

Let us assume for a moment that both parties trust each other and have no privacy concerns with regards to sharing their data. We denote Alice's audio information by $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ and we assume that Bob has K songs the k-th of which we denote by $\mathbf{y}^k = \{y_1^k, y_2^k, \ldots, y_{T^k}^k\}$. In this case Alice can send over a snippet of $\mathbf{x}$ to Bob, Bob can cross-correlate that with all $\mathbf{y}^k$ and return to her the information of the best match. More formally:

*This work was done during an internship at Mitsubishi Electric Research Laboratories

- Alice sends $\mathbf{x}$ to Bob and for every $k = \{1, 2, \ldots, K\}$, Bob computes

$$c_n^k = \sum_r x_r y_{n+r}^k \quad (1)$$

  Let $\mathbf{c}^k$ denote the entire cross-correlation.

- Alice obtains vector $\mathbf{C}$ of K elements, where the k-th element C_k is equal to the maximum value of $\mathbf{c}^k$.

- The index of the maximum element in vector $\mathbf{C}$ indicates the index of Alice's song in Bob's catalog. Alice accesses his catalog and gets the relevant information.

Now let us consider the fact that Alice and Bob have privacy constraints. Alice does not want to send $\mathbf{x}$ to Bob and likewise Bob doesn't want to share $\mathbf{y}^k$ with Alice. This means that the cross-correlation step cannot be completed. Even if this was somehow possible using a trusted third party, Alice would still have to obtain the data she seeks from Bob in which case Bob would know what song Alice has in her possession. In the remainder of this paper we will show how Alice and Bob can complete this collaboration and also satisfy their privacy constraints using SMC.

3. BACKGROUND

Before we tackle the problem, we briefly present some background ideas from the cryptography literature. First, we introduce the concept of Secure Two-party Computations. We then introduce the concept of a *Homomorphic Cryptosystem* which plays a fundamental role in the proposed method.

3.1. Secure two-party computations

The secure application that we present is a specific example of a *secure two-party computation*. Consider the case where Alice and Bob have private data $\mathbf{a}$ and $\mathbf{b}$ respectively and they want to compute the result of a function $f(\mathbf{a}, \mathbf{b})$. Consider a trusted third-party who can take the private data, compute the result $\mathbf{c} = f(\mathbf{a}, \mathbf{b})$, and intimate the result to the parties. Any protocol that implements an algorithm to calculate $f(\mathbf{a}, \mathbf{b})$ is said to be *secure* only if it leaks no more information about $\mathbf{a}$ and $\mathbf{b}$ than what one can gain from learning the result $\mathbf{c}$ from the trusted third-party. We assume a *semi-honest* model for the parties where they follow the protocol but could be saving messages and intermediate results to learn more about other's private data.

The general strategy that we use to create a secure version of an algorithm is as follows:

- we express every step of the algorithm in terms of a handful of basic operations (henceforth called as *primitives*) for which secure implementations are already known, and

- we distribute intermediate results randomly between the two parties such that neither party has access to the entire result. For example, instead of obtaining the result z of a certain step, the parties receive *random additive shares* z_1 and z_2 ($z_1 + z_2 = z$).

The second step ensures that neither party can work his/her way back to private data by utilizing intermediate results. Based on how the primitives are implemented, one can achieve different levels of security and computational/communication efficiency.

3.2. Homomorphic public-key cryptosystem

A public-key cryptosystem is a triple (GE, EN, DE) of probabilistuc polynomial time algorithms for key-generation, encryption and decryption respectively.

- The key-generation algorithm GE generates a valid pair (sk, pk) of private and public keys.

- The encryption algorithm EN, given a public key pk and an input (or *plaintext*) m , outputs $EN(m, pk)$, the *ciphertext* (encryption) of m.

- The decryption algorithm DE, given the private key sk and encrypted text $EN(m, pk)$, returns the original input m.

A cryptosystem is called *homomorphic* if one can indirectly perform specific algebraic operations on the unencrypted data by manipulating the ciphertext. One particular case of homomorphic encryption which we will employ is using the property:

$$EN(a, pk) \times EN(b, pk) = EN(a + b, pk). \quad (2)$$

This implies that a party can add encrypted plaintexts by doing simple computations with ciphertexts, *without having the secret key*. We exploit this property for the protocols we propose in the next section. The Paillier Cryptosystem [6] is an example of one such semantically secure[1] homomorphic cryptosystem.

4. SECURE MUSIC COMPARISON

In this section, we describe how Alice and Bob can interact and perform cooperative computations that enable Alice to identify her song without either party disclosing private data.

[1] Semantic Security, in simple terms, implies that it must be infeasible to derive information about a message given only its ciphertext and the public encryption key. This requirement is equivalent to *ciphertext indistinguishability*, which means an adversary will be unable to distinguish whether a pair of ciphertexts encode the same message or two different messages.

To start, we should identify steps in the algorithm for which secure primitives are available. From Section 2, we know that there are three steps involved. The first step involves the computation of cross-correlations, the second step involves finding the value of the maximum element in a vector and the last step involves finding the *index* of the maximum element in a vector. We shall take one step at a time.

4.1. Step 1: Cross-correlating the music signals

The first step involves finding the cross-correlation between Alice's music snippet $\mathbf{x}$ and each of Bob's songs $\mathbf{y}^k$, $k = \{1, \ldots, K\}$. Now, consider the k-th computation given by equation (1). Alice and Bob would like to perform this computation without disclosing their vectors and this is where the homomorphic cryptosystem plays a role. We use the idea proposed in [7].

1. Alice generates a private and public key pair (sk, pk), and sends the public key to Bob.
2. Alice encrypts each sample of her time series and sends it to Bob. Formally, Alice sends $e_t = EN(x_t, pk)$ to Bob for $t \in \{1, \ldots, T\}$.
3. Consider the computation of c_n^k - the n-th element of the cross-correlation.
   - Bob sets $z \leftarrow \prod_{t=1}^{T} e_t^{y_{n+t}^k}$. Notice that due to the homomorphic property, z represents the encrypted value of c_n^k.
   - Bob generates a random number b_n^k and sends $z' = z \times EN(-b_n^k, pk)$ to Alice. b_n^k will become Bob's share of the result of the dot product.
   - Alice decrypts z' with her private key to obtain her share of the result i.e. she computes $a_n^k = DE(z', sk) = \sum_t x_t y_{n+t}^k - b_n^k$.
4. After the calculation of all elements, Alice and Bob have vectors $\mathbf{a}^k$ and $\mathbf{b}^k$ such that $\mathbf{a}^k + \mathbf{b}^k = \mathbf{c}^k$.

Alice and Bob repeat items (3) and (4) of the above list for all the K song clips of Bob so that at the end, each party has K vectors.

4.2. Step 2: Obtaining the cross-correlation peaks

The second step is to find peaks in the computed cross-correlation vectors. But instead of one vector $\mathbf{c}^k$, the results are distributed as private additive shares $\mathbf{a}^k$ and $\mathbf{b}^k$. Notice that $c_i^k \geq c_j^k \iff (a_i^k - a_j^k) \geq (b_j^k - b_i^k)$. Alice and Bob can do such pairwise comparisons and mimic any standard maximum finding algorithms to learn the value (as additive shares) of the maximum. To perform the comparisons securely, they can use a protocol for Yao's millionaire problem [1, 8].

Once they find the shares corresponding to the peak for all of Bob's songs, they construct two new K-vectors $\mathbf{A}$ and $\mathbf{B}$. Alice assigns a_i^k as the k-th element of $\mathbf{A}$ and Bob assigns b_i^k as the k-th elements of $\mathbf{B}$, where i is the index of the peak. Notice that at this stage, the sum of k-th elements of $\mathbf{A}$ and $\mathbf{B}$ is equal to the peak value of the cross-correlation between $\mathbf{x}$ and $\mathbf{y}^k$. And we have arrived at this stage without Alice or Bob learning anything about the other's data.

4.3. Step 3: Finding the most likely song index

Alice's song corresponds to the index for which the sum of $\mathbf{A}$ and $\mathbf{B}$ is the highest. And we have an additional constraint that Alice doesn't want Bob to know the result of this step. Again, we exploit the property of homomorphic encryption.

We use the concept of a *permute protocol* proposed in [9]. The protocol enables Alice and Bob to obtain additive shares, $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, of a permutation of the vector $\mathbf{A} + \mathbf{B}$, $\pi(\mathbf{A} + \mathbf{B})$, where π is chosen by Alice and Bob has no knowledge of the permutation π. The idea is for Alice to send $\bar{\mathbf{A}} - R$ to Bob, where R is a random number chosen by her. Bob sends back the index of the maximum element of $\bar{\mathbf{A}} + \bar{\mathbf{B}} - R$ to Alice who then computes the real index using the inverse of the permutation π. Since Bob does not know the permutation π, he does not learn the index of the song which Alice has, preserving her privacy. The random number R chosen by Alice guarantees that Bob does not learn the value of the maximum.

The essence of this strategy - the permute protocol - is based on homomorphic encryption and can be described as follows. Given vectors $\mathbf{A}$ and $\mathbf{B}$, and permutation π known only to Alice, the desired result is additive shares $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ such that $\bar{\mathbf{A}} + \bar{\mathbf{B}} = \pi(\mathbf{A}) + \pi(\mathbf{B})$.

- Bob generates a private and public key pair (sk, pk), and sends the public key to Alice.
- Bob encrypts each element of $\mathbf{B}$ using his public key pk and sends the resulting vector to Alice.
- Alice generates a random vector $\mathbf{S}$ and computes a new vector $\boldsymbol{\theta}$ where $\theta_i = EN(B_i, pk)EN(S_i, pk)$, for $i = 1, \ldots, K$. The homomorphic property implies that θ_i is the encryption of $B_i + S_i$.
- Alice permutes $\boldsymbol{\theta}$ and sends $\pi(\boldsymbol{\theta})$ to Bob. Bob decrypts the vector using private key sk to obtain $\bar{\mathbf{B}}$. This implies that $\bar{\mathbf{B}}$ is a permutation of the vector $\mathbf{B} + \mathbf{S}$. Bob cannot try to compare this with the original vector $\mathbf{B}$ to work out permutation since $\mathbf{S}$ is a random vector chosen by Alice that he doesn't know.
- Alice computes $\mathbf{A} - \mathbf{S}$ and then permutes it using π to obtain $\bar{\mathbf{A}} = \pi(\mathbf{A} - \mathbf{S})$.

4.4. Final Step: Obtaining the desired information

Once Alice knows the index, it is quite straightforward for her to learn information about the song if we assume that Bob's catalog is publicly available. She can check the catalog for the index she is interested in and obtains the relevant information. If Bob's catalog is private and he doesn't want Alice to know its entire contents, we will have to use a cryptographic primitive called *oblivious transfer* [10].

Let σ denote the index of the song, known only to Alice. Let Bob's catalog be encoded as a vector $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_K\}$. Alice should receive μ_σ from the database and Bob should not get to know what was transferred. Below is an overview of one [11] of the several cryptographic protocols proposed for this application.

- Alice generates a homomorphic secret/public key pair (sk, pk) and sends pk to Bob. She encrypts σ and sends $z = EN(\sigma, pk)$ to Bob.
- For $k = \{1, \ldots, K\}$, Bob chooses a random r_k, computes $d_k \leftarrow EN(\mu_k, pk) \cdot (z \cdot EN(-k, pk))^{r_k}$, and sends d_k to Alice. Notice that this is equal to the encryption of $\mu_k + r_k(\sigma - k)$.
- Alice decrypts d_σ to obtain μ_σ.

5. DISCUSSION AND CONCLUSIONS

In this paper we have shown how a simple privacy-preserving database lookup can be performed using cryptographic principles. Although this is a simple and illustrative computational model of how one can perform this particular task, extending it to more complex song matching algorithms is very easy. Most signal processing operations eventually break down to dot products and other simple numerical operations which are extensively covered in the SMC literature. These SMC algorithms are readily available and are easy to combine to form complex processing chains. They offer various levels of protection and features which are valuable for many different kinds of computations. As an example a detailed description of a more elaborate privacy-preserving audio application is shown in [12]. It is our hope that as we observe a rising need for privacy-preserving signal computations, such principles will be more widely used in order to alleviate any privacy concerns, and to facilitate the development of more online services relating to collaborative signal processing.

6. REFERENCES

[1] A. C.-C. Yao, "Protocols for secure computation," in *Proc. of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 160–164.

[2] J. Vaidya and C. Clifton, "Privacy preserving k-means clustering over vertically partitioned data," in *ACM SIGKDD Conf on Knowledge Discovery and Data Mining*, 2003.

[3] E. Kiltz, G. Leander, and J. Malone-Lee, "Secure computation of the mean and related statistics," in *Proceedings of the Theory of Cryptography Conference*, ser. Lecture Notes in Computer Science, vol. 3378, 2005, pp. 283–302.

[4] S. Avidan and M. Butman, "Blind vision," in *ECCV*, 2006.

[5] O. Goldreich, "Secure multi-party computation," Working Draft, 2000. [Online]. Available: citeseer.ist.psu.edu/goldreich98secure.html

[6] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of Advances in Cryptology - EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592, 1999, pp. 223–238.

[7] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On private scalar product computation for privacy-preserving data mining," in *Intl. Conference on Information Security and Cryptology*, ser. Lecture Notes in Computer Science, C. Park and S. Chee, Eds., vol. 2506, 2004, pp. 104–120.

[8] H.-Y. Lin and W.-G. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Proc of Intl. Conf. on Applied Cryptography and Network Security*, ser. LNCS, vol. 3531. Springer-Verlag, 2005, pp. 456–466.

[9] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," in *Proceedings of Workshop on Privacy in the Electronic Society*, Washington, DC, USA, October 2003.

[10] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 245–254.

[11] W. Aiello, Y. Ishai, and O. Reingold, "Priced oblivious transfer: How to sell digital goods," in *Advances in Cryptology - EUROCRYPT*, 2001.

[12] P. Smaragdis and M. Shashanka, "A framework for secure speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 4, 2007.