PRIVACY PRESERVING DISTANCE COMPUTATION USING SOMEWHAT-TRUSTED THIRD PARTIES

Abelino Jimenez and Bhiksha Raj

Carnegie Mellon University, Pittsburgh, PA, USA
{abjimenez,bhiksha}@cmu.edu

ABSTRACT

A critically important component of most signal processing procedures is that of computing the distance between signals. In multi-party processing applications where these signals belong to different parties, this introduces privacy challenges. The signals may themselves be private, and the parties to the computation may not be willing to expose them. Solutions proposed to the problem in the literature generally invoke homomorphic encryption schemes, secure multi-party computation, or other cryptographic methods which introduce significant computational complexity into the proceedings, often to the point of making more complex computations requiring repeated computations unfeasible. Other solutions invoke third parties, making unrealistic assumptions about their trustworthiness.

In this paper we propose an alternate approach, also based on third party computation, but without assuming as much trust in the third party. Individual participants to the computation "secure" their data through a proposed secure hashing scheme with shared keys, prior to sharing it with the third party. The hashing ensures that the third party cannot recover any information about the individual signals or their statistics, either from analysis of individual computations or their long-term aggregate patterns. We provide theoretical proof of these properties and empirical demonstration of the feasibility of the computation.

Index Terms— Secure Distance Computation, Information-theoretic Privacy, Locality Sensitive Hashing

.

1. INTRODUCTION

A key signal processing operation is the computation of the distance between two signals. This simple operation can suddenly become challenging if the two signals belong to two mistrustful parties who are unwilling to share them. We must now facilitate the operations, without exposing their signals to one another.

More formally stated, two parties, Alice and Bob, have two real valued vectors $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ respectively. The signal processing operation requires the computation of $\|\mathbf{x}_1 - \mathbf{x}_2\|$, which may be required by one or both parties. However, at the same time, it is required that none besides Alice learns about $\mathbf{x}_1$, and similarly none besides Bob is exposed to $\mathbf{x}_2$.

A number of methods have been suggested in the literature to solve this problem. These have generally involved the use of secure multi-party computation protocols (GCS) [1] or fully homomorphic [2, 3] which enables the computation of distances over encrypted data. Although this provides the participants with the desired privacy, they are computationally unattractive. They can increase both the communication and the computational cost of computing the distance by several orders of magnitude.

An alternate approach uses *trusted third parties* to facilitate the computation. Here, Alice and Bob interact with a third party, Charlie, to compute the distances between their signals with information-theoretic privacy. No party gets additional information despite their computational power. Under this approach, protocols based on secret sharing using polynomials on finite fields have been proposed to compute the euclidean distance [4] between privately held real-valued vectors. While secure, the precision of the computation depends on how well real numbers are represented in the selected field. Moreover, the communication complexity of this kind of protocol is proportional to the dimensionality of the data, which is problematic when Alice and Bob need to compute a distance between high dimensional vectors in presence of communication constraints.

Even if the above concerns were somehow allayed, an additional problem arises from the fundamental definition of the basic problem itself. The outcome of the computation – the distance – itself reveals information about the signals. The party who obtains this result, which may be either Alice or Bob, or both of them, learns about the relationship between their signals. In trusted third party settings, Charlie may get to know this distance too. Ideally, while this "leakage" of information cannot be avoided, we would like to limit it. In particular, it is preferable that the third party Charlie is not exposed to the information at all.

In this paper we propose several third party based protocols for the computation of distances between signals, which address the above issues. We propose to encode the signals through a *modular* hashing scheme [7] with cryptographic properties that ensure that none of the parties can recover the original signal from the hashes, without the appropriate keys. The hashes have the property that they permit *limited* release of the information between vectors – the true distance is only revealed if the vectors are sufficiently close and the distance between them lies below a threshold, but is naturally effectively thresholded when the distance is greater, ensuring that even the parties that *do* get the distance only get as much as is required for effective computation of the signal processing processes, but not enough to chart out the other person's data. We impose protocols on top of the hashing which ensure that the third party cannot gather any information besides the limited distance computed. When the third party may be trusted to know this limited distance then, unlike other cryptographic or secret sharing based schemes, the communication overhead may be made independent of the dimension of the signals, permitting control of the precision of the distance estimate. Finally, we also analyze how the protocols may be modified in order to not expose even the limited distance to the third party.

In the next section (Section 2) we discuss the hashing scheme. In Section 3 we describe our protocols and in Section 4 we explain how they may be extended to hide the distance from the third party, Charlie. Finally we present our conclusions in Section 5.

2. LIMITED DISTANCE COMPUTATION THROUGH MODULAR HASHES

Our proposed approach requires Alice and Bob to transform their signals into hashes prior to further processing. We now describe the transform. This transformation may be considered as a *modular* variant of the p-stable Locality Sensitive Hashing (LSH) [5] [6]. However, unlike the standard LSH, the introduction of modularity helps to generate *uninformative hashes*, i.e. the distribution of the resulting hash does not depend on its input. Furthermore, the euclidean distance between two signals may be estimated through comparison of their hashes, provided the distance is shorter than some threshold.

2.1. Modular Hashing

Definition: Let k be an integer larger than 1, A be an $M \times N$ matrix, $U \in [0,k]^M$, and $\mathbb{Z}_k$ be the set of integers $\{0, 1, \ldots, k-1\}$. We define a *Modular Hash* as the function $Q_{k,A,U} : \mathbb{R}^N \to \mathbb{Z}_k^M$ as

$$Q_{k,A,U}(\mathbf{x}) = \lfloor A\mathbf{x} + U \rfloor \pmod{k} \tag{1}$$

where the floor function and modulo are component-wise.

Definition: If A and U are randomly selected, we say that $Q_{k,A,U}$ is a *Secure Modular Hash* (SMH) if

1. $\forall \mathbf{x} \in \mathbb{R}^N$, $Q_{k,A,U}(\mathbf{x})_i$ is independent of $Q_{k,A,U}(\mathbf{x})_j$ for every $i \neq j$,
2. $\forall i \in \{1, ..., M\}, \forall z \in \mathbb{Z}_k, \mathbb{P}\left(Q_{k,A,U}(\mathbf{x})_i = z\right) = \frac{1}{k}$

One example of SMH is given in [7] as follows:

Theorem 1. $Q_{k,A,U}$ is a SMH if A is randomly generated where its components are i.i.d with

$$a_{ij} \sim \mathcal{N}\left(0, \delta^{-2}\right)$$

for some δ, and U is independent of A, with i.i.d components

$$u_i \sim \text{Unif}(0, k)$$

It is shown in [7] that for SMH defined as above, the expected value of the Hamming distance between $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$ relates to the actual value of $\|\mathbf{x}_1 - \mathbf{x}_2\|$, until the former achieves the value of $1 - \frac{1}{k}$, beyond which the hash is information theoretically secure and reveals no information about $\|\mathbf{x}_1 - \mathbf{x}_2\|$. We will henceforth assume that we work with this kind of SMH.

2.2. Limited Euclidean Distance from Secure Modular Hashes

The *Lee* distance [8] between two integers $a, b \in \mathbb{Z}_k$ is the length of the shortest path from a to b along a ring of circumference k.

$$d_{Lee}(a, b) = \min\left(|a - b|, k - |a - b|\right)$$

Using this distance metric, we can obtain the following result:

Theorem 2. $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^N$, $\forall i \in \{1, ..., M\}$, $\forall k$ even, then we obtain the following relation between the i^{th} components of $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$:

$$\mathbb{E}\left[d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1)_i, Q_{k,A,U}(\mathbf{x}_2)_i\right)\right] = \frac{k}{4} - \frac{2k}{\pi^2}\sum_{j=1}^{\infty}\frac{1}{(2j-1)^2}e^{-2\left(\frac{\pi\|\mathbf{x}_1 - \mathbf{x}_2\|(2j-1)}{\delta k}\right)^2}$$

Moreover, bounding this expression it is possible to prove that when $\|\mathbf{x}_1 - \mathbf{x}_2\| \to \infty$ the described series converges to $\frac{k}{4}$.

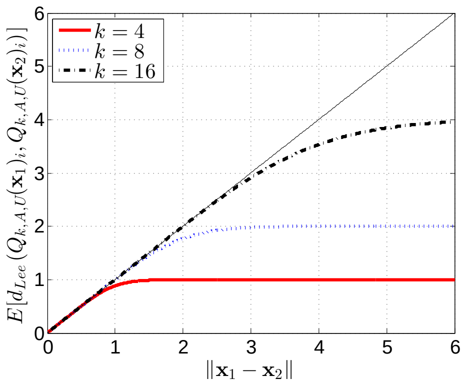

Fig. 1. Expected value of the Lee Distance between $Q_{k,A,U}(\mathbf{x}_1)_i$ and $Q_{k,A,U}(\mathbf{x}_2)_i$ as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|$ using expression given by Theorem 2 and considering $\delta = \sqrt{\frac{2}{\pi}}$.

The *expected* value of the Lee distance between $Q_{k,A,U}(\mathbf{x}_1)_i$ and $Q_{k,A,U}(\mathbf{x}_2)_i$ approximates the euclidean distance between $\mathbf{x}_1$ and $\mathbf{x}_2$. The following result bounds the error of the approximation.

Proposition 1. Let $\delta = \sqrt{\frac{2}{\pi}}$. We define the error

$$\varepsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) := \left|\mathbb{E}[d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1)_i, Q_{k,A,U}(\mathbf{x}_2)_i\right)] - \|\mathbf{x}_1 - \mathbf{x}_2\|\right|$$

The following relation holds:

$$\varepsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) \leq F(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) \tag{2}$$

where

$$F(t, k) = t \cdot \exp\left(-\frac{k^2}{4\pi t^2}\right)$$

It is easy to see that F is increasing in t and decreasing in k. Even more, for fixed t, when k tends to infinity $F(t,k)$ tends to 0. Therefore, we can prove the following proposition:

Proposition 2. $\forall \epsilon > 0$, $\forall T > 0$, $\exists k$ even, $\forall \|\mathbf{x}_1 - \mathbf{x}_2\| < T$

$$\varepsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) < \epsilon$$

This proposition says that given a threshold T we can find a value of k large enough, such that the difference between $\|\mathbf{x}_1 - \mathbf{x}_2\|$ and the expected value of the Lee distance between the corresponding hashes is as small as we want for all $\|\mathbf{x}_1 - \mathbf{x}_2\| < T$. This is illustrated by Figure 1 which shows the expression given by Theorem 2 against $\|\mathbf{x}_1 - \mathbf{x}_2\|$. The relation is seen to be an identity for distances smaller than a threshold, and then converging to $\frac{k}{4}$. Hence, we can compute an accurate estimate of the euclidean distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ directly from the Lee distance between their hashes.

The above relations are all statements of statistical expectation. In real applications, however, we must deal with a single realization of this kind of random process. To proceed, we note that the random variables $\left\{d_{Lee}(Q_{x,A,U}(\mathbf{x}_2)_i, Q_{x,A,U}(\mathbf{x}_2)_i)\right\}_i$ are i.i.d.

We define the *Mean Lee Distance* as the empirical average of the Lee distances between components of $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$:

$$\overline{d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1), Q_{k,A,U}(\mathbf{x}_2)\right)} = \frac{1}{M}\sum_{i=1}^{M} d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1)_i, Q_{k,A,U}(\mathbf{x}_2)_i\right)$$

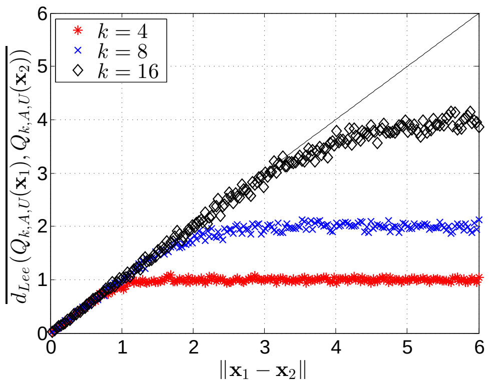

Fig. 2. Mean Lee distance between hashes as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|$ for $\delta = \sqrt{\frac{2}{\pi}}$, $N = 5000$ and $M = 500$.

To determine the value of M such that the Mean Lee Distance between the hashes of $\mathbf{x}_1$ and $\mathbf{x}_2$ is a reasonable estimate of the *expected* value of the Lee distance, we can use the Hoeffding inequality and note that the Lee distance is a number between 0 and $\frac{k}{2}$ to prove that

Proposition 3. For $\delta = \sqrt{\frac{2}{\pi}}$, if $M \geq \frac{\log(2)\cdot(\beta+1)\cdot k^2}{8\epsilon^2}$, then

$$\mathbb{P}\left(\left|\overline{d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1), Q_{k,A,U}(\mathbf{x}_2)\right)} - \mathbb{E}\left[d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1)_i, Q_{k,A,U}(\mathbf{x}_2)\right)_i\right]\right| < \epsilon\right) \geq 1 - \frac{1}{2^\beta}$$

Thus, the Mean Lee Distance is similar to its expected value with high probability if M satisfies the described condition. For instance, if $\beta = 10$, $\epsilon = 0.5$ and $k = 8$, then with $M \geq 244$ we can obtain an estimate of the expected Lee Distance with precision 0.5 with probability more than 0.999.

An important consequence of this result is that the value of M required to obtain a good estimate of the expectation of the Mean Lee Distance between the hashes of $\mathbf{x}_1$ and $\mathbf{x}_2$, which in turn relates to the euclidean distance between the signals, is not dependent on N, the dimensionality of the signal! Thus, we may even consider this kind of transform as a dimensionality-reduction technique or as a dimensionality-hiding hash. Figure 2 shows simulation plots of how the Mean Lee Distance approximates the euclidean distance, when the latter is lower than some threshold, for different values of k. This is absolutely consistent with the previous theoretical results.

3. PROTOCOL TO COMPUTE DISTANCES

We now describe a three-party protocol which uses SMH to estimate the distance between two private signals without revealing them.

Input: Alice and Bob have $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ respectively.
Output: Alice and/or Bob obtain an estimation of $\|\mathbf{x}_1 - \mathbf{x}_2\|$, provided it is lower than some threshold.
Protocol:

1. Alice and Bob agree on a threshold T and precision ϵ, and compute k and M using results from propositions 2 and 3.
2. Alice generates (k, A, U) and sends them to Bob securely, this is

$$a_{ij} \sim \mathcal{N}\left(0, \frac{\pi}{2}\right) \quad \text{and} \quad u_i \sim \text{Unif}(0, k)$$

3. Alice computes $Q_{k,A,U}(\mathbf{x}_1)$ and sends it to Charlie.
4. Bob computes $Q_{k,A,U}(\mathbf{x}_2)$ and sends it to Charles.
5. Charles computes $d = \overline{d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1), Q_{k,A,U}(\mathbf{x}_2)\right)}$ and sends d to Alice and Bob.

First, we can see that after this protocol no party gets anything more than the estimate of $\|\mathbf{x}_1 - \mathbf{x}_2\|$. Alice and Bob never see each others' vectors. Charlie never sees the plain vectors and receives just two vectors in $\mathbb{Z}_k^M$, where each one can be seen as a sequence of independent realizations of draws from a uniform distribution in $\mathbb{Z}_k$. Since Charles does not know (A, U), he does not have any mechanism to reconstruct the actual vectors, and even extract any kind of information more than the distance between them.

Note that it is important that Charlie must not know (A, U); otherwise he may reconstruct the original vectors, particularly if he has some knowledge of the domain of the signals. For the same reason Alice and Bob cannot directly share their hashes after sharing the key (k, A, U). Provided these conditions are followed, the scheme can be shown to be information theoretically secure.

One drawback of this protocol is related to the key transmission. Indeed, if the vectors have a high dimension, transmitting the matrix A may cause a communication overhead. However, we can make A public because the security property of the hash resides on the random vector U. Hence, we can adapt the protocol as follows:

Input: Alice and Bob have $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ respectively. The $N \times M$ matrix A is public.
Output: Alice and/or Bob obtain an estimation of $\|\mathbf{x}_1 - \mathbf{x}_2\|$, provided it is lower than some threshold.
Protocol:

1. Alice generates a M-dimensional vector U with $u_i \sim \text{Unif}(0, k)$, and a random permutation of M components $\mathcal{P}$. Then Alice sends $(U, \mathcal{P})$ to Bob.
2. Alice computes $\mathcal{P}(Q_{k,A,U}(\mathbf{x}_1))$ privately and sends it to Charles.
3. Bob computes $\mathcal{P}(Q_{k,A,U}(\mathbf{x}_2))$ privately and sends it to Charles.
4. Charles computes $d = \overline{d_{Lee}\left(Q_{k,A,U}(\mathbf{x}_1), Q_{k,A,U}(\mathbf{x}_2)\right)}$ and sends d to Alice and Bob.

Note that, despite the fact that vector U results in hashes with uniform distribution, we added the permutation $\mathcal{P}$ in order to obstruct an exhaustive reconstruction by exploring all possible values of U. Following a similar analysis to the first protocol we can conclude that this protocol is secure.

However, these protocols have the disadvantage that the third party has access to the estimate of the euclidean distance between $\mathbf{x}_1$ and $\mathbf{x}_2$. Ideally we would like not to trust the third party to know this. In the next section we explore some solutions for this setting.

4. PROTOCOL TO HIDE DISTANCE VALUE

In order to not reveal the distance to the third party, we propose two protocols. The first does not consider the third party, and the distance computation between two real valued vectors is based on a Secure Hamming Distance computation between binary vectors.

The second protocol considers the idea of adding noise to the hashes in order to hide the true distance value.

4.1. No Third Party

One way to prevent the use of a third party is to establish a cryptographic protocol which lets us compute the Lee Distance privately.

However, the computation of the Lee Distance involves a minimum between two numbers, a task that can increase the computational complexity.

Nevertheless, it is possible to reduce the Lee distance computation to a Hamming Distance computation. In fact, we can encode any element in $\mathbb{Z}_k$ as a vector in $\{0,1\}^{\frac{k}{2}}$. In particular, if $a \in \mathbb{Z}_k$, we define $c(a) \in \{0,1\}^{\frac{k}{2}}$ as follows:
If $a \leq \frac{k}{2}$,

$$c(a)_i = \begin{cases} 1 & \text{if } i \leq a \\ 0 & \text{otherwise} \end{cases}$$

If $a > \frac{k}{2}$,

$$c(a)_i = \begin{cases} 0 & \text{if } i \leq a - \frac{k}{2} \\ 1 & \text{otherwise} \end{cases}$$

For example, in $\mathbb{Z}_6$ the coding left

$$c(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad c(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad c(2) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$c(3) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad c(4) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad c(5) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

This kind of coding has the property that

$$d_{Lee}(a,b) = d_{Hamming}(c(a)\,,c(b)) = \sum_{i=1}^{k/2} c(a)_i \oplus c(b)_i$$

Therefore, an element $\mathbf{z} \in \mathbb{Z}_k^M$ can be coded as $c(\mathbf{z}) \in \{0,1\}^{M\cdot\frac{k}{2}}$, where

$$c(\mathbf{z})^\top = \begin{bmatrix} c(z_1)^\top, & c(z_2)^\top, & \cdots, & c(z_M)^\top \end{bmatrix}$$

Then, the Mean Lee Distance between $\mathbf{z}_1$ and $\mathbf{z}_2 \in \mathbb{Z}_k^M$ is equal to

$$\frac{d_{Hamming}(c(\mathbf{z}_1),\, c(\mathbf{z}_2))}{M}$$

With this result we enable the estimation of the euclidean distance between two points in $\mathbb{R}^N$ using the Hamming distance of two binary vectors. As a consequence, we can replace the third party computation by any protocol which computes securely the Hamming distance between binary vectors; for example, [9] defines a two party protocol based on Oblivious Transfer.

To summarize, we define the following protocol,

Protocol:

1. Alice generates (k, A, U) and sends them to Bob.
2. Alice computes $c(Q_{k,A,U}(\mathbf{x}_1))$ privately.
3. Bob computes $c(Q_{k,A,U}(\mathbf{x}_2))$ privately.
4. Alice and Bob apply a secure two party protocol to compute the Hamming distance d between $c(Q_{k,A,U}(\mathbf{x}_1))$ and $c(Q_{k,A,U}(\mathbf{x}_2))$.
5. Alice and Bob compute the estimate of $\|\mathbf{x}_1 - \mathbf{x}_1\|$ as $\frac{d}{M}$.

Unlike most protocols based on homomorphic encryption to compute the euclidean distance, the complexity of the presented protocols does not depend on the dimensionality of the vector at the moment of applying the cryptographic technique. Hence, our proposal for computing the euclidean distance between two vectors is to embed them in binary vectors and compute the Hamming distance between them.

One problem of this protocol is the fact we lose information-theoretical privacy, unlike the previous ones.

4.2. Obfuscating Information to the Third Party

One alternative is to retain the third party but hide information from it through obfuscation. We simply propose adding noise to the corresponding hashes. The protocol is as follows:

Protocol:

1. Alice generates (k, A, U), two independent vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ in $\mathbb{Z}_k^P$, where each component is independent and distributed uniformly in $\mathbb{Z}_k$, and a permutation $\mathcal{P}$ of $M + P$ elements, and sends $(k, A, U, \mathbf{z}_1, \mathbf{z}_2, \mathcal{P})$ to Bob.
2. Alice computes $\mathcal{P}\left(\begin{bmatrix} Q_{k,A,U}(\mathbf{x}_1)) \\ \mathbf{z}_1 \end{bmatrix}\right)$ privately, and sends it to Charlie.
3. Bob computes $\mathcal{P}\left(\begin{bmatrix} Q_{k,A,U}(\mathbf{x}_2)) \\ \mathbf{z}_2 \end{bmatrix}\right)$ privately, and sends it to Charlie.
4. Charlie computes the Mean Lee distance d between the received vectors and sends it to Alice and Bob.
5. Alice and Bob compute privately the Mean Lee distance $\tilde{d}$ between $\mathbf{z}_1$ and $\mathbf{z}_2$ and compute

$$\frac{(M+P)\cdot d - P \cdot \tilde{d}}{M}$$

obtaining the estimate of $\|\mathbf{x}_1 - \mathbf{x}_2\|$.

It is easy to see that the Mean Lee distance between $Q_{k,A,U}(\mathbf{x}_1))$ and $Q_{k,A,U}(\mathbf{x}_2))$ is equal to the last expression of the protocol. One of the advantages of this protocol is the fact that, when P is large enough, $\mathcal{P}\left(\begin{bmatrix} Q_{k,A,U}(\mathbf{x}_2)) \\ \mathbf{z}_2 \end{bmatrix}\right)$ is indistinguishable from a vector with uniformly distributed i.i.d components, therefore, the value of d should be close to $\frac{k}{4}$ for every pair $\mathbf{x}_1$ and $\mathbf{x}_2$.

A natural drawback is that the number of parameters to run the protocol, which may cause a computational overhead.

5. CONCLUSIONS

In this paper we have presented a random transformation, that given a threshold T, can generate hashes from vectors that preserve the euclidean distance between them if it is smaller than T. These hashes are uninformative if the random parameters of the function are unknown.

With this kind of transformation, we are able to describe protocols to compute distances privately and efficiently. In fact, the proposed transformation is not only uninformative, it can be seen as a transformation which reduces the data dimension and preserves the euclidean distance through the Lee distance output space.

Although the fact that the distance is preserved until some threshold seems to be a drawback, if both Alice and Bob know the maximum possible distance between their vectors, then they can set an appropriate value of k which lets preserve distances to any desired threshold. We also describe how to hide the distance value from the third party, action that may increase the computational cost of the protocol, as it is expected.

We still have many questions. How can we extend these protocols to multiparty versions? How easy is a reconstruction of $\mathbf{x}$ knowing $Q_{k,A,U}(\mathbf{x})$ and (k, A, U)? We have also trusted the third party to not collude with Alice or Bob. Can this restriction be lifted?

The proofs of the presented theorems can be found in:
`https://arxiv.org/abs/1609.05178`

6. REFERENCES

[1] M. Pathak and B. Raj, “Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models.” IEEE Transactions on Audio, Speech and Language Processing, Vol 21:2, pp. 397-406, 2013.

[2] M. Naehrig, K. Lauter and V. Vaikuntanathan, “Can homomorphic encryption be practical?.” Proceedings of the 3rd ACM workshop on Cloud computing security workshop, 2011.

[3] S. Rane and P. T. Boufounos, “Privacy-Preserving Nearest Neighbor Methods: Comparing Signals Without Revealing Them.” IEEE Signal Processing Magazine, Vol. 30(2), pp. 18-28, 2013.

[4] Y. Wang, P. Ishwar and S. Rane, “Information-theoretically secure three-party computation with one active adversary.” 2012. [Online]. Available: http://arxiv.org/abs/1206.2669

[5] P. Indyk and R. Motwani. “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality.” Proceedings of 30th Symposium on Theory of Computing, 1998.

[6] M. Datar, N. Immorlica, P. Indyk and V.S. Mirrokni, “Locality-Sensitive Hashing Scheme Based on p-Stable Distributions.” Proceedings of the Symposium on Computational Geometry, 2004.

[7] A. Jimenez, B. Raj, J. Portelo and I. Trancoso, “Secure Modular Hashing.” Proceedings of WIFS, 2015.

[8] E. Deza and M. Deza, “Dictionary of Distances”, Elsevier, 2014.

[9] J. Bringer, H. Chabanne and A. Patey, “SHADE: Secure HAmming DistancE Computation from Oblivious Transfer.” Financial Cryptography and Data Security Volume 7862 of the series Lecture Notes in Computer Science pp 164-176. 2013.