Privacy-Preserving Nearest Neighbor Methods

Shantanu Rane and Petros Boufounos

Comparing two signals is one of the most essential and prevalent tasks in signal processing. A large number of applications fundamentally rely on determining the answers to the following two questions: (1) How should two signals be compared? (2) Given a set of signals and a query signal, which signals are the nearest neighbors of the query signal, i.e., which signals in the database are most similar to the query signal?

The nearest neighbor (NN) search problem is defined as follows: Given a set $\mathcal{S}$ containing points in a metric space $\mathcal{M}$, and a query point $\mathbf{x} \in \mathcal{M}$, find the point in $\mathcal{S}$ that is closest to $\mathbf{x}$. The problem can be extended to K-NN, i.e., determining the K nearest neighbors of $\mathbf{x}$. In this context, the 'points' in question are signals, such as images, videos or other waveforms. The qualifier 'closest' refers to a distance metric, such as the Euclidean distance or Manhattan distance between pairs of points in $\mathcal{S}$. Finding the nearest neighbor of the query point should be at most linear in the database size and is a well-studied problem in conventional NN settings.

However, this problem much less straightforward when the signals under consideration are private, i.e., they cannot be revealed to the party conducting the NN search. At first glance, the problem appears to be a non-starter when privacy constraints are imposed: Is it even possible to find the distance between two signals without knowing what the signals are? Is it possible to determine the minimum of these distances without knowledge of the distances themselves? Fortunately, the answer to both these questions is affirmative. The intent of this article is to

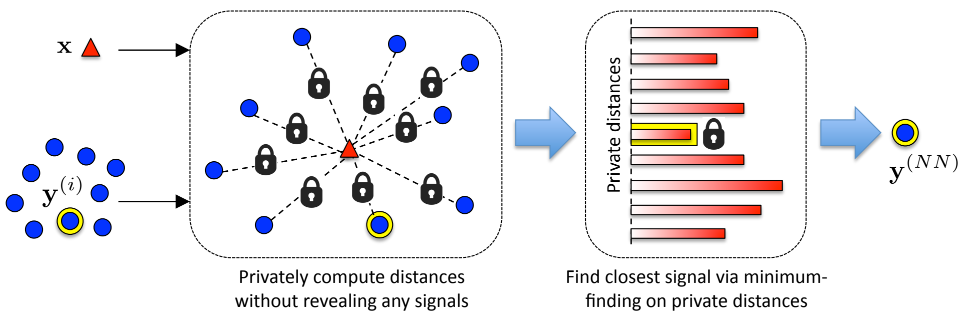
$\mathbf{x}$
$\mathbf{y}^{(i)}$
Privately compute distances without revealing any signals
Private distances
Find closest signal via minimum-finding on private distances
$\mathbf{y}^{(NN)}$

Fig. 1. A PPNN protocol can be broken down into two privacy-preserving protocols.

provide a tutorial survey of the methods that have been developed to answer these questions, i.e., to solve the NN search problem under a variety of privacy constraints.

While thinking of a Privacy-Preserving Nearest Neighbor (PPNN) method, the reader will find it convenient to divide it into two distinct problems: a method for Privacy-preserving (PP) distance computation followed by a method for PP minimum finding, as shown in Fig. 1. PPNN problems arise repeatedly in many of today's emerging applications: secure biometric authentication [1, 2], privacy-preserving face recognition [3, 4], private recommender systems [5], privacy-preserving speech processing [6], and many others.

The choice of mathematical tools used for PPNN, as well as the structure and complexity of the resulting protocols are dictated by the privacy model under consideration. These models encapsulate assumptions such as the privacy requirements, the behavior of participating entities and the possibility of information sharing among the participants.

The privacy guarantee can be based on the assumption that certain problems in number theory, e.g., factorization of large numbers, are intractable for an adversary with limited computational resources. It is important to note that many of these assumptions are hitherto unproven, and that a computational privacy guarantee may be vulnerable to a quantum computer in the future. A more powerful privacy guarantee, based on information theory, ensures that an adversary cannot compromise privacy of any party without knowing a specific key, even if he has unlimited computational resources. In this article, we cover examples of both kinds of cryptographic tools, i.e., we consider both *computationally private* and *information-theoretically private* protocols.

The parties in a privacy-preserving protocol may behave in a *honest but curious* fashion, i.e., they may follow the rules of the protocol, but can use all the information made available to them by the protocol and try to guess the data held by other parties. A more severe assumption on the behavior of the parties is that they may be *malicious*, i.e., they may arbitrarily deviate from the rules of the protocol with the aim of discovering data held by other parties. Furthermore, two or more parties in a protocol may choose to *collude* with the goal of forcing the protocol to generate a wrong result, or to compromise the data owned by an unsuspecting party. Naturally, it is more difficult to design protocols that are secure against colluders and malicious parties.

In this article, we will concentrate primarily on honest but curious parties—also called *semi-honest* parties—for two reasons: Firstly, much of the recent surge of secure signal processing research has considered semi-honest, non-colluding parties. Secondly, our goal with this paper is to provide a clear understanding of the basic methods and tools for PPNN protocol design.

Many of these methods can later be strengthened to accommodate malicious adversaries via verifiable secret sharing and zero knowledge proofs, if the steep increase in protocol complexity is acceptable.

The remainder of this article is organized as follows: In the next section, we describe PPNN tools and protocols in which privacy guarantees are based on computationally intractable problems. Following this section, we describe PPNN tools and protocols with information-theoretic privacy. Then, we discuss the important role that dimensionality-reducing embeddings play in PPNN, either by reducing the complexity of previously discussed cryptographic protocols, or by providing privacy guarantees of their own. Towards the end of the article, we briefly discuss the design of protocols resistant to malicious colluders and discuss the most interesting open problems in the field.

I. PPNN WITH COMPUTATIONAL PRIVACY

The first class of PPNN methods that we describe is the class of computationally private NN protocols which leverages several interesting advances made in cryptography during the past two decades. The characteristic feature of these methods is that signals are kept private from other parties by means of encryption, and computations needed for the two tasks shown in Fig. 1 are performed in the encrypted domain. Thus, the first task is to compute encrypted distances between a pair of signals held by mutually untrusting parties, e.g., a client with a query signal $\mathbf{x}$ and a server with a database signal $\mathbf{y}^{(i)}$, as shown in Fig. 2. The second task is, given a set of encrypted distances at the server, to find which one is the smallest, thus providing the nearest neighbor $\mathbf{y}^{(NN)}$. Below, we describe various tools for computationally private multiparty computation—homomorphic cryptosystems, oblivious transfer, and garbled circuits—that are used to accomplish these two tasks.

A. *Tools for Computational Privacy*

The principal cryptographic tool to perform encrypted-domain distance computation is a public-key homomorphic cryptosystem. A *homomorphic* cryptosystem has the property that the encryption of a function of some variables can be computed by performing operations on the encryptions of the variables. For example, in an *additively* homomorphic cryptosystem, encryption of the sum of two variables is computed simply by multiplying individual encrypted variables. More precisely, denoting the encryption function by $E(\cdot)$, an additively homomorphic cryptosystem [7–9] satisfies, for integers a, b,:

$$E(a) \cdot E(b) = E(a+b) \text{ and } (E(a))^b = E(ab)$$

Several other flavors of homomorphic cryptosystems have been designed, based on computationally intractable problems such as inverting a discrete logarithm, determining N-residues modulo N^2, factorizing large numbers, etc. In a *multiplicatively* homomorphic cryptosystem [10], encryption of the product of plain text values is computed by multiplying individual encrypted values. Yet another scheme allows the encryption of a quadratic function of the plaintexts to be computed from individual encrypted values [11]. One of the most spectacular results in cryptography in recent times has been a constructive proof of the existence of *fully* homomorphic encryption [12] which enables computation of encryptions of arbitrary polynomial functions of several variables. Fully homomorphic cryptosystems are based on integer lattices and are currently very complex to implement, but their simplification for practical usage is an extremely active research area today. For more information, the reader is referred to [13] and a companion article in this issue [14].

We will adopt additively homomorphic cryptosystems in our explanation because they have been among the first cryptosystems leveraged for PPNN [15, 16]. Furthermore, while the number-theoretic properties that enable additive homomorphisms are somewhat involved, the operations on the ciphertexts are straightforward enough to convey an understanding of the typical data manipulation steps in cryptographic PPNN protocols. In the remainder of this section, we assume that the client and the server in Fig. 2 use a public key additively homomorphic cryptosystem.

Given a set of encrypted distances at a server, selecting the nearest neighbor involves first finding the smallest of these distances, and then transferring the data point corresponding to the smallest distance to the querying client. Performing these tasks requires a cryptographic primitive protocol known as *Oblivious Transfer* (OT) [17], a technique that dates back to the beginnings of secure computation research. Suppose Alice has a bit $b \in \{0, 1\}$ and Bob has data (m_0, m_1). Then 1-out-of-2 OT, the most basic kind of OT, accomplishes the following: At the end of the protocol, Alice receives m_b, while not discovering $m_{\bar{b}}$. Bob does not discover the value of b, thus he does not know which element of his vector has been revealed to Alice. Since OT is one of the most fundamental protocols in secure computation—even beyond our PPNN problem—we present an example implementation of 1-out-of 2 OT in Sidebar 1, based on homomorphic

Inputs: Alice has $b \in \{0,1\}$, Bob has (m_0, m_1).

Output: Alice obtains m_b, Bob obtains nothing.

Protocol: OT can be implemented with an additively homomorphic cryptosystem as follows:

1) Alice generates homomorphic encryptions $E(\bar{b})$ and $E(b)$ and sends them to Bob.
2) Bob cannot decrypt these cipher texts. He computes $E(\bar{b})^{m_0} E(b)^{m_1} = E(\bar{b}m_0 + bm_1)$ and sends it to Alice.
3) Alice decrypts Bob's transmission to obtain $\bar{b}m_0 + bm_1$, which is precisely equal to the desired m_b.

The aim above was to show that the OT problem is indeed solvable using the number-theoretic properties of cryptosystems. Many other implementations of OT are possible, notably using traditional public-key cryptography and garbled circuits.

Sidebar 1: Implementations of Oblivious Transfer.

encryption. 1-out-of-2 OT and its generalization to k-out-of-n OT are used when the client wants to obliviously recover nearest neighbors without discovering the far neighbors stored at the server.

It has been shown that OT is a sufficient primitive for secure function computation, i.e., if we can construct an algebraic circuit whose inputs correspond to the data owned by mutually untrusting parties, and whose output corresponds to the function of interest, then OT can be used to compute the function while preserving all privacy constraints [18, 19]. In principle, this can be accomplished by executing a 1-out-of-2 OT for every wire of the circuit. More pertinent to our subject, OT can be used to construct a "secure millionaire" protocol [17]: given two inputs a and b held by untrusting parties, the protocol reveals which party has the greater value, without revealing the individual values. The secure millionaire protocol, or some variant of it, is ubiquitous whenever comparisons, maximum-finding or minimum-finding must be carried out under privacy constraints.

B. Binary Hamming Distance Computation

Suppose that $\mathbf{x}, \mathbf{y}$ are length-n binary vectors owned by the client and server respectively. The goal is for the server to compute the encryption of the *binary Hamming distance* given by $d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} x_i \oplus y_i$, where $\oplus$ is the binary XOR operation. In this and the following examples, assume that both client and server possess a public encryption key for an additively homomorphic cryptosystem, however, only the client possesses the private decryption key, i.e.,

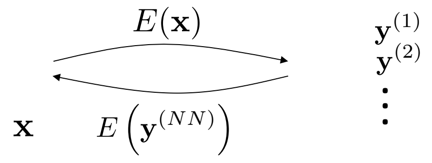

Fig. 2. A thin client interacts with a server using a cryptographic protocol to find the signal in the server's database that is closest to its own signal.

the protocols are based on the public/private key-pair of the client. To begin, the client transmits n elementwise encryptions $E(x_i)$ to the server. To compute the encrypted Hamming distance, the server computes

$$\prod_{i=1}^{n} E(x_i)E(y_i)E(x_i)^{-2y_i} = \prod_{i=1}^{n} E(x_i + y_i - 2x_i y_i) = \prod_{i=1}^{n} E(x_i \oplus y_i) = E\left(\sum_{i=1}^{n} x_i \oplus y_i\right)$$

which is the desired result. Note that the server need only compute the first expression using the encryptions $E(x_i)$ received from the client, encryptions $E(y_i)$ calculated at the server, and plaintext values of y_i that are available to it. The additive homomorphic property ensures that the result of that computation is the encrypted Hamming distance.

C. Squared Euclidean Distance Computation

Making a small change from the above, suppose that $\mathbf{x}, \mathbf{y}$ are length-n integer vectors owned by the client and server respectively. The goal is for the server to compute the encryption of the *squared Euclidean distance*, which is given by $d_E(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n}(x_i - y_i)^2$. Again, as explained above, the client transmits n elementwise encryptions $E(x_i)$ to the server. In addition, it also transmits the encrypted summation $E(\sum_{i=1}^{n} x_i^2)$ to the server, which then computes

$$E\left(\sum_{i=1}^{n} x_i^2\right) E\left(\sum_{i=1}^{n} y_i^2\right) \prod_{i=1}^{n} E(x_i)^{-2y_i} = E\left(\sum_{i=1}^{n}(x_i - y_i)^2\right)$$

which is the desired distance. It is easy to extend this method to compute a *weighted* version of this distance measure. Thus, for a public weight vector $\mathbf{w}$, a distance of the form $d_E(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \sum_{i=1}^{n} w_i(x_i - y_i)^2$ can be computed privately using homomorphic properties.

D. Edit Distance

So far, we have restricted $\mathbf{x}$ and $\mathbf{y}$ to have equal lengths, but encrypted-domain computation techniques based on homomorphic cryptosystems are powerful enough to tackle distances between

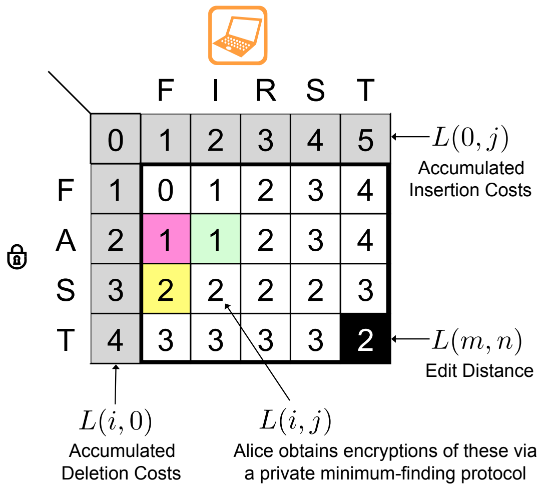
F I R S T
F A S T
0 1 2 3 4 5
1 0 1 2 3 4
2 1 1 2 3 4
3 2 2 2 2 3
4 3 3 3 3 2
$L(0,j)$
Accumulated Insertion Costs
$L(m,n)$
Edit Distance
$L(i,0)$
Accumulated Deletion Costs
$L(i,j)$
Alice obtains encryptions of these via a private minimum-finding protocol

Fig. 3. An example of edit distance computation, in which "FAST" is one of strings owned by the server and "FIRST" is a string owned by a querying client. In this asymmetric setup, only the client possesses a decryption key for an additively homomorphic cryptosystem. The server computes encryptions of $L(i,j)$ via an interactive privacy-preserving protocol.

sequences of unequal lengths. Suppose the client possesses a m-length sequence $\mathbf{x}$ and the server possesses an n-length sequence $\mathbf{y}$, where each element of the sequences belongs to a finite alphabet set $\mathcal{A}$. Examples of $\mathcal{A}$ include $\{0,1\}$, $\{a,b,...,y,z\}$ and so on. Let $I(\alpha)$ denote the cost of inserting a symbol α into a sequence and $D(\alpha)$ denote the cost of deleting a symbol α from the sequence. Finally, let $S(\alpha,\beta)$ denote the cost of substituting a symbol α with another symbol β. A combination of insertions, deletions and substitutions can transform $\mathbf{x}$ into $\mathbf{y}$. Several such combinations are possible in general, and each combination incurs an aggregate cost. The *edit distance* or *Levenshtein distance* is the minimum aggregate cost necessary to perform this transformation [20].

We use $L(i,j)$ to denote the edit distance between the two subsequences $x_1x_2...x_i$ and $y_1y_2...y_j$. Fix $L(0,0)=0$. Define accumulated insertion and deletion costs as

$$L(0,j)=\sum_{k=1}^{j} I(x_k) \quad \text{and} \quad L(i,0)=\sum_{k=1}^{i} D(y_k)$$

In the example of Fig. 3, the insertion and deletion cost of each symbol is 1, so the accumulated costs just increase monotonically in the topmost row and the leftmost column. Mathematically, the edit distance $L(m, n)$ is given by the following recurrence relation for $1 \leq i \leq m, 1 \leq j \leq n$:

$$L(i, j) = \min \left\{ \begin{array}{c} L(i-1, j) + D(y_j), \\ L(i, j-1) + I(x_i), \\ L(i-1, j-1) + S(x_i, y_j) \end{array} \right\} \tag{1}$$

As before, the goal is for the server to compute $E(L(m, n))$ without discovering $\mathbf{x}$ and without revealing $\mathbf{y}$ to the client. This is accomplished by filling up an $m \times n$ matrix by privately computing $E(L(i, j))$ at each stage according to (1). Observe that, while calculating $E(L(i, j))$ and traversing the matrix in a raster fashion, the server already possesses previously computed encryptions of $L(i-1, j), L(i, j-1)$ and $L(i-1, j-1)$. It also knows $I(x_i)$ and can obtain $E(D(x_i))$ from the client. Now, the three terms in the curly brackets in (1) represent just additions, hence their encryptions are computable with additive homomorphic encryption, provided an encryption of the substitution costs $S(\cdot, \cdot)$ is available [21]. In general, the substitution cost $S(x_i, y_i)$ can be expressed as a look-up table indexed by the sequence element x_i of the client and y_i of the server. Therefore, an encryption of $S(x_i, y_i)$ is always computable via oblivious transfer.

Then, it remains to compute $E(L(i, j))$—the encryption of the minimum of the three quantities which are themselves encrypted. This is accomplished via a minimum-finding protocol. Methods to accomplish this are described in the next subsection. Repeating this process mn times, the server obtains $E(L(m, n))$ which is the desired result. The privacy-preserving edit distance protocol, which essentially is an instance of dynamic programming with privacy, can also be used with minor modifications for related distance metrics such as sequence similarity and Earth-mover's distance [22].

E. Privacy-Preserving Minimum Finding

Having obtained encrypted distances $E(d(\mathbf{x}, \mathbf{y}^{(i)}))$ of a query signal from several signals in a database, the next step is to identify the closest one, i.e., to privately find $\mathbf{y}^{(NN)}$ such that $d(\mathbf{x}, \mathbf{y}^{(NN)}) \leq d(\mathbf{x}, \mathbf{y}^{(i)})$ for all i. Given that some privacy-preserving distance computation has been carried out prior to minimum finding, it is vital that the minimum finding protocol should not reveal the index $i^* = \arg\min_i d(\mathbf{x}, \mathbf{y}^{(i)})$ of the nearest neighbor to the client or to the server. In some applications, it may be necessary to also conceal the value of $d(\mathbf{x}, \mathbf{y}^{(NN)})$ from the server.

One approach to accomplish the above goals is to first apply one or more "blind-and-permute" steps which obfuscate the ordering of the distances from one or both parties [21]. This is followed by a series of secure pairwise comparisons—achieved by millionaire protocols—that can be efficiently executed using garbled circuits. Sidebar 2 presents an example protocol, in which "blind-and-permute" steps are accomplished using homomorphic encryption, after which millionaire protocols are used for minimum-finding.

Rather than using homomorphic encryption for minimum finding, an alternative method is to use garbled circuits [24, 25]. The idea is to construct an efficient secure millionaire protocol using garbled circuits, and then to compose a sequence of millionaire protocols to find the minimum distance. Recently, a garbled circuit construction was proposed in which secure evaluation of XOR gates is essentially free, requiring only one XOR operation on secret keys [24]. This construction has been leveraged for the millionaire protocol, by designing a garbled circuit for secure comparison that consists predominantly of XOR gates [25]. The method has approximately half of the protocol overhead incurred by previous methods employing garbled circuits. In some cases, the overhead of this method is even lower than that of the homomorphic cryptosystem-based protocol described above.

Here, we have focused on communicating the key ideas in encrypted-domain signal processing for PPNN by considering protocols for semi-honest non-colluding parties. To prove that privacy is preserved in such protocols, it is sufficient to verify that decryption is possible only at the party that possesses the private decryption key, that inputs from other parties are made inaccessible to the decrypting party by means of additive blinding and—in the case of minimum finding—permutations. For more extensive discussion on computationally private methods, which includes privacy proofs, complexity studies, possible attacks and collusion scenarios and emerging applications, we refer the reader to a recently published survey [26].

F. Extensions to Multiparty Scenarios

To illuminate the basic operations in encrypted-domain protocols, the scenario above was deliberately kept simple. The above techniques can be extended in several beneficial directions. For example, it may be too restrictive to require that the server possess plaintext data $\mathbf{y}^{(i)}$ in Fig. 2. What if the untrusted server is merely used for outsourced computation and storage of data owned by other private parties, as shown in Fig. 4? To perform distance computation in this scenario, there are at least two possible alternative strategies: One strategy is to allow calculation of the cross terms, such as $E(x_i)^{-2y_i}$ in the Hamming and Euclidean distance calculation at the private parties. The cost of this strategy is an increase in computational complexity at the

Inputs: Client has the vector $\mathbf{x}$. The server has vectors $\mathbf{y}^{(i)}$ and encrypted distances, $E(d(\mathbf{x}, \mathbf{y}^{(i)}))$. Following our convention, we denote the n-length vector of distances by the symbol $\mathbf{d}$, and its elementwise homomorphic encryption by $E(\mathbf{d})$. Assume that there is a second additively homomorphic cryptosystem whose decryption key is known to the server alone. To distinguish this from the client's cryptosystem, we will denote the encryption function for this second cryptosystem by $E'(\cdot)$.

Output: Client obtains $\mathbf{y}^{(NN)}$, i.e., the vector $\mathbf{y}^{(i)}$ which minimizes $d(\mathbf{x}, \mathbf{y}^{(i)})$.

Protocol: The minimum finding protocol proceeds thus:

1) The server generates a random vector $\mathbf{s}$ and a permutation π_S on $\{1, 2, ..., n\}$. It computes $E(\pi_S(\mathbf{d} - \mathbf{s}))$ and sends it to the client. It also sends $E'(\pi_S(\mathbf{s}))$ to the client.
2) The client decrypts $\pi_S(\mathbf{d} - \mathbf{s})$. Then it generates a random vector $\mathbf{c}$ and a permutation π_C on $\{1, 2, ..., n\}$. The client then computes $E'(\pi_C(\pi_S(\mathbf{s}) - \mathbf{c}))$ and sends it to the server, while retaining with itself $\pi_C(\mathbf{c} + \pi_S(\mathbf{d} - \mathbf{s})) =: \mathbf{a}$.
3) The server decrypts $\pi_C(\pi_S(\mathbf{s}) - \mathbf{c}) =: \mathbf{b}$. Notice that the server and client now possess additive shares of the permuted distance vector, i.e., $\mathbf{a} + \mathbf{b} = \pi_C(\pi_S(\mathbf{d}))$, but neither of them can reverse the other's permutations.
4) Further, letting u, v denote the permuted indices, notice that $d_u > d_v \Rightarrow a_u + b_u > a_v + b_v \Rightarrow a_u - a_v > b_v - b_u$. Thus, using the permuted shares, the server and client can perform repeated secure pairwise comparisons using Yao's millionaire protocol [17] or its variants. In this manner, the client obtains the permuted index $\pi_C(\pi_S(i^*))$ of the minimum distance, and hence the permuted index of the nearest neighbor data $\mathbf{y}^{(NN)}$.
5) The client then runs a 1-out-of-n OT to obliviously recover $\mathbf{y}^{(NN)}$ from the server. Other than $\mathbf{y}^{(NN)}$, no other data from the server is revealed to the client. The server neither discovered $\mathbf{x}$ nor the index i^* of the nearest neighbor.

An alternative strategy to steps 4 and 5 above is to homomorphically combine the shares and obtain at the server's end a permuted vector of encrypted distances. Following this, the client and server run an interactive protocol at the end of which each encrypted distance is converted into a vector of encrypted bits. Given an encrypted binary representation, the index corresponding to the minimum distance can be identified using homomorphic computations on the bitplanes [23], or by applying millionaire protocols on the bitplanes.

Sidebar 2: Minimum-finding using blind-and-permute operations and millionaire protocols

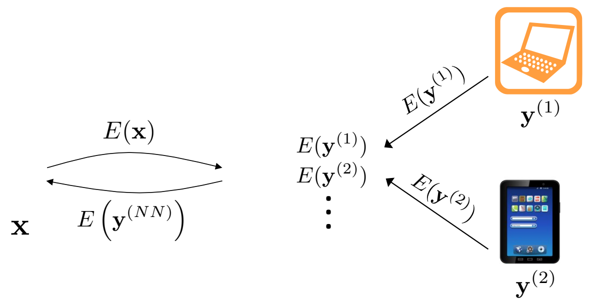

Fig. 4. Cryptographic PPNN protocols can be extended such that several parties outsource storage and computation to the untrusted server. The type of homomorphic cryptosystem used dictates how the protocol complexity is divided among the parties and the server.

individual participants, and the need for more bandwidth to transmit the encryptions to and from the server. Another strategy is to use homomorphic cryptosystems that enable computation of quadratic expressions or polynomials [11, 12]. This second method allows the data owners to ship their encryptions to the untrusted server, and to fully outsource the task of distance computation.

So far, we have considered PPNN with computational methods for star-connected configurations, i.e., the server accepts and processes encrypted inputs from untrusting participants. Next, we consider a more general setup in which there are several mutually untrusting parties, some of which possess private data, and with some pairs of parties having communication channels available to them. A simple example of this setup is shown in Fig. 5. In this multiparty computation scenario, the two untrusting parties possessing data $\mathbf{x}$ and $\mathbf{y}$ can communicate with each other as well as with the server. In such a scenario it is certainly possible for the parties to encrypt their data and construct computationally private NN protocols using the techniques discussed above. However, passing ciphertexts back and forth on the pairwise communication channels typically incurs a huge communication overhead. Fortunately, when inter-party communication channels are available, protocol design can be dramatically simplified using information theoretic tools rather than encryption. We consider these methods in the next section.

II. PPNN with Information-Theoretic Privacy

The second class of PPNN methods we describe is information theoretically private NN protocols which leverage algebraic tools that have classically been applied in information theory and error correction coding. The characteristic feature of these methods is the generation of shared secrets which are sent over the communication links according a specified function computation protocol. The primary advantage of these protocols is that they provide privacy against *computationally unbounded* adversaries. From a practical point of view, as there is no encryption being performed, the size of the data sent over the communication links is much smaller than that of the ciphertexts in the previous section. Thus, this class of methods is well-suited for, but not limited to, multiparty computation scenarios in which several untrusting parties are connected to other parties via pairwise communication links. Below, we describe various tools for information theoretically private multiparty computation—random pads, secret sharing and common randomness—that can be used to accomplish the two tasks of Fig. 1.

A. Tools for Information-Theoretic Privacy

For the scenario of multiple connected parties explained above, there are powerful constructive results for privacy-preserving function computations based on secret sharing using polynomials on finite fields [27–30]. These results can be used when the functions are distances between signals held by the untrusting participants. It is useful to think of the secure computation procedure as a 3-step process: In the first step, each participant generates shares of his input by evaluating a polynomial at several points over a finite field. These shares—each of which is independent of the true input—are then distributed to the other participants. The second step involves each party performing local computations using the shares it possesses, such that it obtains a share of the desired function value. A slight caveat here is that when the local computations involve multiplication, the parties must monitor the degree of the product polynomials and, if the degree becomes too high, they require a secure degree-reduction step which does involve a small amount of inter-party communication [27]. The final step is a resolution step in which one or more designated parties can combine the shares and obtain the function value.

We again illustrate private distance computation using Hamming distance and Euclidean distance as examples. For clarity of exposition, we use the multiparty computation scenario in Fig. 5, in which Alice and Bob can communicate with each other, as well as with the server. In the first example, we illustrate the use of random pads and permutations for private computation of distance measures on binary data. These distances can indeed be computed using secret sharing, but our aim with the first example is to point out that, in some cases, PPNN is possible using familiar information-theoretic tools. In the second example, we use the polynomial secret sharing schema explained above to privately compute pairwise Euclidean distances between signals owned by the untrusting parties.

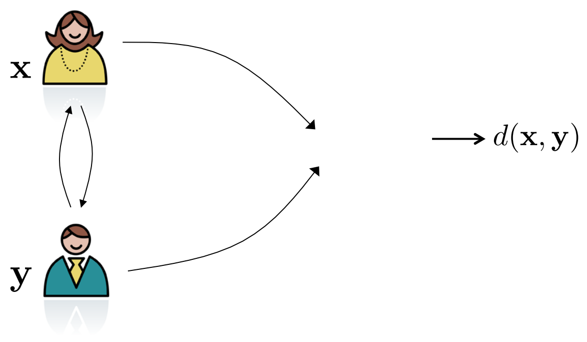

Fig. 5. Mutiple parties can interact with a server to compute distances (or additive shares of distances) with information-theoretic privacy. The parties can transmit the shares to each other using error-free secure links.

B. Binary Hamming Distance

Alice and Bob own n-length binary vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. Alice randomly generates a binary string $R_1, \ldots, R_n \sim$ i.i.d. Bernoulli$(1/2)$ and randomly chooses π, a permutation of $\{1, \ldots, n\}$, uniformly and independently of $\mathbf{x}$. She securely sends $\mathbf{R}$ and π to Bob. Then Alice and Bob respectively send $\pi(\mathbf{x} \oplus \mathbf{R})$ and $\pi(\mathbf{y} \oplus \mathbf{R})$ to the server. The server just adds the messages received from Alice and Bob, and the Hamming weight of the summation gives the desired distance. Notice that this protocol is information-theoretically private in the sense that no party can discover any information about inputs possessed by the other parties. Interestingly, this rather simple protocol remains privacy preserving even when one of the three participants becomes malicious [31].

C. Squared Euclidean Distance

Alice and Bob own n-length integer vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. Then, classical secure multiparty computation techniques [27, 28] based on secret sharing [29] may be used to compute several distance measures that are polynomials in x_i, y_i. As an example, Sidebar 3 considers the PP computation of squared Euclidean distance [31].

In the example considered, the untrusted third party discovers the distances between signals, which may reveal too much information in some PPNN applications. In that case, the protocols can easily be modified so that one or more parties only obtain additive shares of the distances.

The following protocol ensures the privacy-preserving computation of squared Euclidean distance among two untrusting parties, Alice and Bob, using a third party, the server [31].

1) Alice creates polynomials $p_i(j) = \alpha_i j + x_i$ for $i = 1, 2, ..., n$, where the α_i are uniformly distributed over $\{1, 2, ..., N\}$. She evaluates the polynomials at three values, keeping $p_i(1)$ for herself, sending $p_i(2)$ to Bob, and $p_i(3)$ to the server.
2) Similarly, Bob creates n polynomials $q_i(j) = \beta_i j + y_i$. He stores $q_i(2)$ for himself, sends $q_i(1)$ to Alice, and $q_i(3)$ to the server.
3) Consider the squared distance polynomial $r(j) = \sum_{i=1}^{n}(p_i(j) - q_i(j))^2$. Notice that $r(0) = \sum_{i=1}^{n}(x_i - y_i)^2$. The server computes $r(3)$, receives $r(1)$ from Alice and $r(2)$ from Bob.
4) The server reconstructs the degree-two polynomial r using interpolation from the three points $r(1)$, $r(2)$, and $r(3)$. Then, it evaluates $r(0)$ which is the desired squared distance.

Sidebar 3: Protocol for privacy-preserving squared Euclidian distance

D. Minimum Finding

An important consideration in the practical feasibility of information-theoretically PPNN is that, while distance measures can be computed with perfect privacy as seen above, minimum finding is a challenge. From the section on computationally private PPNN, it is clear that minimum finding must involve signal comparison at some stage. Secure signal comparison, i.e., the millionaire problem, requires OT, which cannot be realized from scratch, i.e., with protocols that use only noiseless channels and randomness locally available at the untrusting parties [18, 19]. On the other hand, in the presence of *correlated* randomness, such as a noisy communication channel or a distributed random source available between pairs of parties, OT can indeed be performed with unconditional privacy [32–35]. Under these conditions, the minimum finding task described in the previous section could conceivably be performed with information-theoretic privacy.

III. PPNN WITH RANDOMIZED EMBEDDINGS

Embeddings are very useful tools in the design of privacy-preserving NN cryptosystems. Mathematically, an *embedding* is a transformation of a set of signals in a high-dimensional space to a (typically) lower-dimensional one such that some aspects of the geometry of the set are preserved, as depicted in Fig. 6. This property is beneficial in two ways. First, since the set geometry is preserved, the PPNN protocols discussed above can be performed directly using

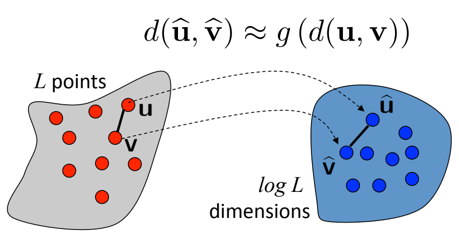
$d(\widehat{\mathbf{u}}, \widehat{\mathbf{v}}) \approx g\left(d(\mathbf{u}, \mathbf{v})\right)$
L points
u
v
û
v̂
log L dimensions

Fig. 6. Distance-preserving embeddings approximately preserve a function of the distance, allowing the NN problem to be solved in a space that (typically) has fewer dimensions. Embeddings can be used in conjunction with cryptographic protocols, and can also be designed to be privacy-preserving on their own.

the low-dimensional embeddings, rather than the underlying signals. As a result, whether the underlying protocol is computationally or information-theoretically private, the computational complexity and the communication overhead are both reduced. Second, certain embeddings can be designed with privacy-preserving properties *of their own*, thereby completely eliminating the need for cryptographic tools and the associated overhead. Thus, some types of embeddings complement the cryptographic methods discussed earlier, whereas other types of embeddings completely replace them. Below, we consider both types.

A. *Johnson-Lindenstrauss Embeddings*

The best known embeddings are the Johnson-Lindenstrauss embeddings [36]—functions $f : \mathcal{S} \to \mathbb{R}^K$ from a finite set of signals $\mathcal{S}$ to a K-dimensional vector space such that given two signals $\mathbf{x}$ and $\mathbf{y}$ in $\mathcal{S}$, their images satisfy:

$$(1-\epsilon)\|\mathbf{x}-\mathbf{y}\|_2^2 \leq \|f(\mathbf{x})-f(\mathbf{y})\|_2^2 \leq (1+\epsilon)\|\mathbf{x}-\mathbf{y}\|_2^2.$$

In other words, these embeddings preserve ℓ_2 distances, i.e., Euclidean distances, of point clouds within a small factor, measured by ϵ.

The original paper by Johnson and Lindenstrauss demonstrated that an embedding preserving the distances as described above exists in a space with dimensionality logarithmic in the number of signals in the embedded set and independent of the dimensionality of those signals. More precisely, an embedding exists in a $K = O(\frac{1}{\epsilon^2}\log L)$-dimensional space, where L is the number of signals in $\mathcal{S}$ (its cardinality) and ϵ the desired tolerance in the embedding. Subsequent work showed that it is straightforward to compute such embeddings using a linear mapping. In particular, the function $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a matrix whose entries are drawn randomly from specific distributions, is a J-L embedding with overwhelming probability. Commonly used distributions are i.i.d. Gaussian, i.i.d. Rademacher[1], or i.i.d. uniform.

A J-L embedding typically results in a significant dimensionality reduction which reduces the complexity of NN computations. Distance preservation and dimensionality reduction ensure that PP distance computation can be achieved at low protocol overhead by applying the methods of the previous two sections to the embeddings $f(\mathbf{x})$ and $f(\mathbf{y})$, rather than using the original signals $\mathbf{x}$ and $\mathbf{y}$.

One important consideration in using such embeddings is the quantization which is necessary before a possible encryption and transmission of the embedded signals. If the quantization is not carefully designed, the performance of the embedding suffers [37]. Furthermore, in the extreme case of quantization down to 1-bit, the embedding can fail to preserve the amplitudes of signals and, therefore, their ℓ_2 distances. It does, however, preserve their angle, i.e., their correlation coefficient.

More precisely, if $f(\mathbf{x}) = \operatorname{sign}(\mathbf{A}\mathbf{x})$, where $\operatorname{sign}(\cdot)$ denotes the element-wise sign function and $\mathbf{A}$ consists of i.i.d. normally distributed entries, the Hamming distance in the embedding space is approximately proportional to the acute angle between the original signals, as measured by the inverse cosine of their inner product [38, 39]:

$$\left| d_H\left(f(\mathbf{x}), f(\mathbf{y})\right) - \frac{1}{\pi} \arccos \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} \right| \leq \epsilon,$$

where $d_H(\cdot, \cdot)$ denotes an appropriately normalized Hamming distance. In applications in which the nearest neighbors are determined using the angle as a distance metric, as opposed to the ℓ_2 distance, it suffices to determine the near neighbors in the embedding domain using Hamming distance as a similarity measure [38].

B. Locality Sensitive Hashing

In the same spirit, Locality Sensitive Hashing (LSH) encodes linear vector spaces into binary spaces [40]. The premise of LSH is that the set of signals of interest can be separated into groups—often referred to as "buckets" in the hashing literature—using a probabilistic mapping such that neighboring signals belong to the same group, i.e., hash to the same value, with very high probability. Thus, one can rapidly search for the near-neighbors by looking for signals in

[1]which is identical to the Bernoulli distribution, except that the variable takes values in $\{-1, +1\}$ instead of $\{0, 1\}$.

the same group. To verify with high probability if two signals are neighbors, one can just verify if their hash is the same.

Mathematically, given two signals $\mathbf{x}$, and $\mathbf{y}$, their hashes are functions $f : \mathcal{S} \to \{0, 1\}^n$ that satisfy

$$f(\mathbf{x}) = f(\mathbf{y}) \text{ w.h.p., if } \|\mathbf{x} - \mathbf{y}\|_2 \leq r$$

$$f(\mathbf{x}) \neq f(\mathbf{y}) \text{ w.h.p., if } \|\mathbf{x} - \mathbf{y}\|_2 \geq cr$$

for some radius of interest r and constant factor $c > 1$. Thus, if two signals have distance smaller than r they will hash to the same value with high probability. Using hashes, one can confidently identify the neighbors of a signal within a distance r, while excluding signals farther than cr.

LSH methods shift the focus to bits instead of dimensionality, making them appealing for PPNN computation, especially in communication-heavy cryptographic protocols [6]. While several LSH approaches exist, the most efficient encoding is based on the Leech lattice [40]. One notable method is based on randomized projections and scalar quantization [41], resembling a quantized Johnson-Lindenstrauss embedding. Thus, in addition to the LSH guarantees, it also offers distance preserving guarantees as described above.

While dimensionality-reducing embeddings of the ℓ_2 distance, as we discuss above, have been most widely studied and applied, we note for the interested reader that research in the past decade has also revealed other, more exotic embeddings—notably, efficient embeddings of edit distance into an ℓ_1 metric space [42–44].

C. Secure Embeddings

The two types of embeddings described above are useful as parts of a rigorous privacy-preserving solution only when combined with a computational or information-theoretic protocol to guarantee privacy, as shown by the combinations of JL embeddings and homomorphic encryption for computing ℓ_1 and ℓ_2 distances [45]. It would be desirable if an embedding could provide security guarantees on its own, without requiring a cryptographic protocol as a wrapper. Such a class of embeddings has been introduced based on recent work in universal scalar quantization [46]. These embeddings provide information-theoretic privacy on their own, which makes them attractive for lightweight PPNN searches.

These embeddings rely on a Johnson-Lindenstrauss style projection, followed by scaling,

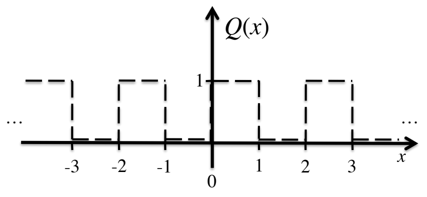

Fig. 7. A non-monotonic quantization such as the above can be used to provide information-theoretic guarantees for privacy-preserving nearest neighbors. This function is equivalent to a multibit quantization, where only the least significant bit is preserved and all other bits are discarded.

dithering and scalar quantization:

$$f(\mathbf{x}) = Q(\Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w})),$$

where $\mathbf{A}$ is a random matrix with normally-distributed, i.i.d. elements, Δ^{-1} is a scaling factor—operating element-wise, in an abuse of notation—$\mathbf{w}$ is the dither, uniformly distributed in $[0, \Delta]$, and $Q(\cdot)$ is the scalar quantizer operating element-wise on the input vector. In order to preserve security and privacy, the projection matrix $\mathbf{A}$ and the dither $\mathbf{w}$ should be treated as a secret key and not be revealed to the untrusted party.

The key feature that makes this embedding privacy preserving is the modified scalar quantizer; it is designed to have non-contiguous quantization intervals, as shown in Fig. 7. The quantizer can be thought of as a regular quantizer, determining a multi-bit representation of a signal and then keeping only the least significant bit (LSB) of that representation. Thus, scalar values in $[2k, 2k+1)$ quantize to 1 and scalar values in $[2k+1, 2(k+1))$, for any k, quantize to 0.

More precisely, the embedding satisfies

$$|d_H\left(f(\mathbf{x}), f(\mathbf{y})\right) - g\left(\|\mathbf{x} - \mathbf{y}\|_2\right)| \leq \epsilon,$$

where $d_H(\cdot, \cdot)$ is the Hamming distance in the embedding space, and $g(d)$ is the distance map shown in Fig. 8. The map is approximately linear for small d and becomes a constant exponentially fast for large d greater than a distance threshold D_0. The slope of the linear section and the distance threshold D_0 are determined by the embedding parameters, Δ and $\mathbf{A}$.

An information-theoretic argument guarantees that these embeddings convey no information about two signals if the distance between the signals is greater than the threshold D_0. For an M-bit embedding, assuming that the embedding matrix and the dither remain private, the mutual

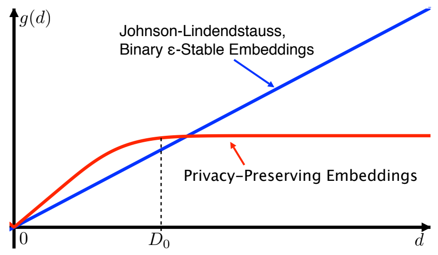

Fig. 8. Embeddings preserve a function $g(d)$ of the original signal distance $d = \|\mathbf{x} - \mathbf{y}\|_2$. For most embeddings, such as Johnson-Lindenstrauss and Binary ϵ-stable Embeddings, this function is linear, as shown in blue. For privacy-preserving embeddings, the function is approximately linear initially and quickly flattens after a certain distance D_0, as shown in red. These embeddings provide information-theoretic security guarantees for signals with distance greater than D_0.

information between the embeddings of two signals decays as

$$I(f(x); f(y)|d) \leq 10Me^{-cd^2},$$

where c is a constant that depends on the embedding parameters. This ensures that an eavesdropper cannot reconstruct the signals and an untrusted third party cannot obtain any information about the relationship between the two signals if they are far apart. This property is very useful in the design of lightweight protocols for PPNN applications.

IV. Open Problems and Discussion

In this article, we have covered three classes of PPNN methods, showing by examples how the building blocks, i.e., distance computation and minimum finding, can be realized under privacy constraints. Table I compares and contrasts these classes. It is our hope that the article will bring this important topic from privacy-preserving computation to a broader signal processing audience, and stimulate new theoretical and applied work at the intersection of signal processing, cryptography and information theory. Our goal was to bring out the basic ideas and primitive operations that form the building blocks of PPNN search. Because of this focus, we have not covered here an important related topic, namely processing signals such that they are in the correct format to be used as inputs to PPNN protocols. In particular, signal processing methods such as media fingerprinting and robust hashing [47–49] often provide some privacy on their own, which complements the PPNN protocol. Some of these methods are covered in other articles in this special issue.

Criterion	Computational Methods	Information-Theoretic Methods	Randomized Embedding Methods
Privacy Guarantee	Based on the unproven hardness of factorization, finding residues, discrete logarithms, lattice problems	Based on information-theoretic privacy wherein the adversary does not know a secret key, or a shared secret	Based variously on information-theoretic privacy, statistical privacy, and the difficulty of matrix inversion.
Adversary	Computationally bounded	Computationally unbounded	Bounded or unbounded
Tools	Homomorphic encryption, garbled circuits	Polynomial secret sharing, random pads, permutations	JL embedding, LSH, secure embeddings
Overhead per $\mathbf{x}, \mathbf{y}$ pair.	$O(n)$ ciphertext additions, multiplications and transmissions.	Communication is linear in the circuit size and polynomial in the number of parties	2 matrix multiplications + $O(\frac{1}{\varepsilon^2}\log L)$
Distance functions $d(\mathbf{x}, \mathbf{y})$	Polynomials in $\mathbf{x}, \mathbf{y}$	Polynomials in $\mathbf{x}, \mathbf{y}$	Hamming, Euclidean, Manhattan, and Edit distances, Angles
Feasibility of minimum finding	Yes	Yes, if correlated randomness is available	No

TABLE I

A HIGH-LEVEL COMPARISON OF THE THREE CLASSES OF PPNN METHODS

There are several fascinating open problems in secure multiparty computation which directly impact the privacy, speed, complexity and versatility of PPNN methods. In the class of cryptographic approaches, advances in doubly homomorphic encryption are especially promising. Specifically, if the ciphertext size and the complexity of the encryption and decryption operations can be made manageable, then it would become feasible for a client to encrypt its data and send it to a cloud-based server, which would return the PPNN result in a single round. In such a truly outsourced computation setup, there would be no need for intermediate exchange and decryption of ciphertexts, thus PPNN protocols would be greatly simplified.

In information-theoretic methods, many classical privacy guarantees [27, 28] for collusions or malicious attacks only apply for computation with more than 3 parties. For example, N-party computation with malicious colluders is possible only when the coalition of colluders has size less than $N/3$. Does this mean that for the 3-party scenario considered in this article, PPNN is impossible even if a single party becomes malicious? Surprisingly, for this 3-party case, it is still unknown if any class of functions—let alone distance functions—can be computed privately in the presence of a malicious adversary.

The area of randomized embeddings is receiving increased attention from the database and compressed sensing communities, and new flavors of distance preservation are constantly being discovered. From the perspective of PPNN search, several exciting open problems remain in the design and analysis of non-invertible distance-preserving embeddings. For example, how difficult is it to recover a signal given its LSH embedding and the embedding matrix? If the LSH embedding is replaced by the secure embedding discussed above, does the problem become NP hard? Can randomized algorithms—similar to ℓ_1-norm minimization problems in compressed sensing—solve the inversion with high probability? Answers to the preceding two questions have implications for the feasibility of very low-complexity, encryption-free protocols based on randomized embeddings.

REFERENCES

[1] M.Barni, T.Bianchi, D.Catalano, M. Raimondo, R. Labati, P.Failla, D.Fiore, R.Lazzeretti, V.Piuri, F.Scotti, and A.Piva, "Privacy-preserving fingercode authentication," in *ACM Workshop on Multimedia and Security (MMSEC2010)*, Rome, Italy, Sept 2010.

[2] M. Blanton and P. Gasti, "Secure and efficient protocols for iris and fingerprint identification," in *European Symposium on Research in Computer Security (ESORICS)*, Leuven, Belgium, September 2011.

[3] A. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *International Conference on Information Security and Cryptology (ICISC '09)*, Seoul, Korea, December 2009.

[4] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, R. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *International Symposium on Privacy Enhancing Technologies (PET)*, Seattle, WA, August 2009, pp. 235–253.

[5] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, "Efficiently Computing Private Recommendations," in *International Conference on Acoustic, Speech and Signal Processing-ICASSP*, Prague, Czech Republic, May 2011, pp. 5864–5867.

[6] M. Pathak and B. Raj, "Privacy-preserving speaker verification as password matching," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.

[7] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Advances in Cryptology, EUROCRYPT 99*, vol. 1592. Springer-Verlag, Lecture Notes in Computer Science, 1999, pp. 233–238.

[8] J. Benaloh, "Dense Probabilistic Encryption," in *Proc. Workshop on Selected Areas of Cryptography*, Kingston, ON, Canada, May 1994, pp. 120–128.

[9] I. Damgård and M. Jurik, "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System," in 4th *Intl. Workshop on Practice and Theory in Public Key Cryptosystems*, Cheju Island, Korea, Feb. 2001, pp. 119–136.

[10] T. El-Gamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, pp. 469–472, 1985.

[11] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography Conference*, 2005, pp. 325–341.

[12] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, March 2010.

[13] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully homomorphic encryption over the integers,” *Advances in Cryptology–EUROCRYPT*, pp. 24–43, 2010.

[14] “Recent advances in fully homomorphic encryption: A possible future for signal processing in the encrypted domain,” *IEEE Signal Processing Magazine*, 2012.

[15] M. Shaneck, Y. Kim, and V. Kumar, “Privacy preserving nearest neighbor search,” in *Proc. of the Sixth IEEE Intl. Conf. Data Mining - Workshops*, Washington, DC, USA, 2006, pp. 541–545.

[16] Y. Qi and M. J. Atallah, “Efficient privacy-preserving k-nearest neighbor search,” *Intl. Conference on Distributed Computing Systems*, vol. 0, pp. 311–319, 2008.

[17] A. C.-C. Yao, “How to Generate and Exchange Secrets,” in *Proc. 27th Annual Symposium on Foundations of Computer Science (SFCS)*. Washington, DC, USA: IEEE Computer Society, 1986, pp. 162–167.

[18] J. Kilian, “Founding cryptography on oblivious transfer,” in *ACM symposium on Theory of computing*. Chicago, IL: ACM, May 1988, pp. 20–31.

[19] R. Cramer and I. Damgård, “Multiparty computation, an introduction,” *Contemporary cryptology*, pp. 41–87, 2005.

[20] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.

[21] M. Atallah, F. Kerschbaum, and W. Du, “Secure and private sequence comparisons,” in *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, Washington, DC, Oct. 2003, pp. 39–44.

[22] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[23] F. Kerschbaum, D. Biswas, and S. de Hoogh, “Performance comparison of secure comparison protocols,” in *Database and Expert Systems Application, 2009. DEXA’09. 20th International Workshop on*. IEEE, 2009, pp. 133–136.

[24] V. Kolesnikov and T. Schneider, “Improved garbled circuit: Free xor gates and applications,” in *International Colloquium on Automata, Languages and Programming (ICALP)*, Reykjavik, Iceland, July 2008, pp. 486–498.

[25] V. Kolesnikov, A. Sadeghi, and T. Schneider, “Improved garbled circuit building blocks and applications to auctions and computing minima,” in *Cryptology and Network Security (CANS)*, Kanazawa, Japan, December 2009, pp. 1–20.

[26] I. Lagendijk, Z. Erkin, and M. Barni, “Encrypted signal processing for privacy protection,” *IEEE Signal Processing Magazine*, January 2013.

[27] M. Ben-Or, S. Goldwasser, and A. Wigderson, “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” in *ACM symposium on Theory of computing*. ACM, 1988, pp. 1–10.

[28] D. Chaum, C. Crépeau, and I. Damgard, “Multiparty unconditionally secure protocols,” in *ACM Symposium on Theory of computing*. ACM, 1988, pp. 11–19.

[29] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[30] G. Asharov and Y. Lindell, “A full proof of the bgw protocol for perfectly-secure multiparty computation,” *Advances in CryptologyCRYPTO 2011*, 2011.

[31] Y. Wang, P. Ishwar, and S. Rane, “Secure three-party computation with one malicious adversary,” *Arxiv preprint arXiv:1206.2669*, 2012.

[32] C. Crépeau, "Efficient cryptographic protocols based on noisy channels," in *Advances in CryptologyEURO-CRYPT97*. Springer, 1997, pp. 306–317.

[33] A. Nascimento and A. Winter, "On the oblivious transfer capacity of noisy correlations," in *IEEE International Symposium on Information Theory (ISIT)*, Seattle, Washington, July 2006, pp. 1871–1875.

[34] R. Ahlswede and I. Csiszár, "On oblivious transfer capacity," in *IEEE International Symposium on Information Theory (ISIT)*, Nice, France, June 2007, pp. 2061–2064.

[35] Y. Wang and P. Ishwar, "Bootstrapped oblivious transfer and secure two-party function computation," in *IEEE International Symposium on Information Theory (ISIT)*. Seoul, Korea: IEEE, June 2009, pp. 1303–1307.

[36] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space, Conference in modern analysis and probability (New Haven, Conn., 1982)," *Contemp. Math*, vol. 26, pp. 189–206, 1984.

[37] M. Li, S. Rane, and P. Boufounos, "Quantized embeddings of scale-invariant image features for mobile augmented reality," in *IEEE International Workshop on Multimedia Signal Processing*, Banff, Canada, September 17-19 2012.

[38] Y. Plan and R. Vershynin, "Dimension reduction by random hyperplane tessellations," *Arxiv preprint arXiv:1111.4452*, 2011.

[39] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors," *Submitted*, April 2011.

[40] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Comm. ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[41] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, ser. SCG '04. New York, NY, USA: ACM, 2004, pp. 253–262.

[42] A. Andoni, M. Deza, A. Gupta, P. Indyk, and S. Raskhodnikova, "Lower bounds for embedding edit distance into normed spaces," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, 2003, pp. 523–526.

[43] Z. Bar-Yossef, T. Jayram, R. Krauthgamer, and R. Kumar, "Approximating edit distance efficiently," in *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*. IEEE, 2004, pp. 550–559.

[44] R. Ostrovsky and Y. Rabani, "Low distortion embeddings for edit distance," *Journal of the ACM (JACM)*, vol. 54, no. 5, p. 23, 2007.

[45] S. Rane, W. Sun, and A. Vetro, "Privacy-Preserving Approximation of L1 Distance for Multimedia Applications," in *IEEE International Conference on Multimedia and Expo (ICME)*, Singapore, July 2010, pp. 492–497.

[46] P. T. Boufounos and S. Rane, "Secure binary embeddings for privacy preserving nearest neighbors," in *Workshop on Information Forensics and Security (WIFS)*, Foz do Iguaçu, Brazil, November 2011.

[47] W. Lu, A. Varna, A. Swaminathan, and M. Wu, "Secure image retrieval through feature protection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Taipei, Taiwan: IEEE, March 2009, pp. 1533–1536.

[48] W. Lu, A. Varna, and M. Wu, "Security analysis for privacy preserving multimedia retrieval," in *IEEE International Conference on Image Processing (ICIP)*, Hong Kong, September 2010, pp. 2093–2096.

[49] W. Lu and M. Wu, "Multimedia forensic hash based on visual words," in *IEEE International Conference on Image Processing (ICIP)*, Hong Kong, September 2010, pp. 989–992.

Author Biographies

Shantanu Rane (Ph.D., Stanford University, 2007) is a Principal Research Scientist at Mitsubishi Electric Research Laboratories in Cambridge, MA. He serves as an Associate Editor of the IEEE Transactions on Information Forensics and Security, the IEEE Signal Processing Letters, and is a member of the IFS Technical Committee. Dr. Rane has co-authored several papers in secure signal processing and distributed source coding, and has participated in standardization activity for H.264/AVC video compression and ISO/SC37 biometrics.

Petros T. Boufounos (D.Sc., MIT, 2006) is a Principal Member of Research Staff at Mitsubishi Electric Research Laboratories in Cambridge, MA and a visiting scholar at the Rice University Electrical and Computer Engineering department in Houston, TX. He is interested in signal acquisition and processing with focus on compressive sensing and data representations. Dr. Boufounos is currently an associate editor at IEEE Signal Processing Letters. He is also a member of the IEEE, Sigma Xi, Eta Kappa Nu and Phi Beta Kappa, and has been an MIT Presidential Fellow.