Privacy-Preserving Collaborative Analytics on Medical Time Series Data

Xiaoning Liu, Yifeng Zheng, Xun Yi, and Surya Nepal

Abstract—Medical time series data analytics based on dynamic time warping (DTW) greatly benefits modern medical research. Driven by the distributed nature of medical data, the collaboration of multiple healthcare institutions is usually necessary for a sound medical conclusion. Among others, a typical use case is disease screening for public health, where multiple healthcare institutions wish to collaboratively detect over their joint datasets the patients whose medical records have similar features to the given query samples. However, sharing the medical data faces critical privacy obstacles with the increasingly strict legal regulations on data privacy. In this paper, we present the design of a novel system enabling privacy-preserving DTW-based analytics on distributed medical time series datasets. Our system is built from a delicate synergy of techniques from both cryptography and data mining domains, where the key idea is to leverage observations on the advancements in plaintext DTW analytics (e.g., clustering and pruning) to facilitate the scalable computation in the ciphertext domain, through our tailored security design. Extensive experiments over real medical time series datasets demonstrate the promising performance of our system, e.g., our system is able to process a secure DTW query computation over 15K time series sequences in 34 minutes.

Index Terms—Privacy preservation, time series analytics, dynamic time warping, secure medical application

1 Introduction

Medical time series data analytics produces insights to facilitate the advancement in various medical fields, such as biomedical research and clinical decision making. Among others, a typical use case is a disease screening for public health, where multiple healthcare institutions (e.g., hospitals) and stakeholders (e.g., researchers) wish to jointly detect patients whose medical records have similar features to given samples. Such an analytic task often needs a matching algorithm specified for medical time series data to produce a reliable conclusion. Dynamic time warping (DTW) is a robust distance metric to match time series data. As long as two time series sequences display similar shapes, they will be matched even they are shifted along the time axis. It has been widely applied for medical time series data analytics, like identifying Premature Ventricular Contraction (PVC) with Electrocardiography (ECG) data [2] and Cardiac Tamponade with Photoplethysmogram (PPG) data [3].

Despite the advantages, the deployment of the DTW-based medical analytics system in practice is heavily impeded due to acute privacy concerns. For most medical practices, the volume and diversity of data accumulated in a single hospital usually cannot provide rich disease information due to the natural distribution of medical data [4], [5]. Thus there is a need for the collaboration among multiple healthcare institutions so as to cater for high quality analytics and realistic demands. However, unauthorized exposure of the confidential medical records will seriously compromise the patients' privacy, and may inflict severe financial losses as the data are proprietary [6], [7]. Meanwhile, these institutions cannot ever share or pool together their sensitive data in the clear because of laws and privacy regulations like HIPAA in USA [8] and GDPR in Europe [9].

To overcome the privacy hurdle, a plausible solution that fits in the paradigm of joint medical data analytics is to utilize generic secure multi-party computation (MPC) techniques [10], [11], [12], [13]. Unfortunately, the direct adoptions would suffer from prohibitively high computation and communication overheads. As such, a recent trend in the latest studies is to develop application-specific MPC protocols [4], [14], [15], [16]. However, to our best knowledge, the design of secure and scalable DTW-based collaborative analytics remains unexplored.

DTW, as a complicated and iterative mining algorithm, is not directly amiable with the MPC techniques. It evaluates the similarity between two time series sequences via a dynamic programming approach. Specifically, this approach needs to iteratively work on portions of the sequences (i.e., subsequences) until the whole sequences are evaluated. In each iteration, it needs to compute and aggregate all pairwise underlying distances between feature vectors of time series sequences as the cumulative distances, and acquire the minimum one as the distance between the current evaluated subsequences. Substantial vector-wise operations lead to high computational complexity that scales quadratically to the sequence length. Such complicated processing is even non-trivial to deal with in the plaintext domain. It is thus quite challenging and requires delicate treatments to design

- *X. Liu and X. Yi with the School of Computer Science and Software Engineering, RMIT University, Melbourne, VIC 3001, Australia. Email: maggie.liu@rmit.edu.au, xun.yi@rmit.edu.au.*
- *Y. Zheng is with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China. This work was done while Y. Zheng was with Data61, CSIRO, Marsfield NSW 2122, Australia, and also with the Cyber Security Cooperative Research Centre (CRC), Joondalup WA 6027, Australia. Email: yifeng.zheng@my.cityu.edu.hk.*
- *S. Nepal is with Data61, CSIRO, Marsfield NSW 2122, Australia, and also with the Cyber Security Cooperative Research Centre (CRC), Joondalup WA 6027, Australia. Email: surya.nepal@data61.csiro.au.*
- *Y. Zheng is the corresponding author.*

A preliminary version [1] of this paper was presented at The European Symposium on Research in Computer Security (ESORICS), 2019.

a system with full support for the healthcare institutions to perform scalable DTW-based medical analytics in the encrypted domain over their joint data.

In light of the above observations, in this paper, we design the first system tailored for privacy-preserving DTW-based collaborative analytics on medical time series data. Our system is built from a delicate synergy of techniques from both cryptography and data mining domains. Such a synergy enables a scalable analytics system with provable security guarantees, addressing the above requirements: embracing large-scale medical time series data that are naturally decentralized, conducting practical DTW-query computation with guaranteed security, and producing reliable medical conclusion by securely harnessing joint data.

Our first insight is to design a hybrid pruning strategy which leverages observations on effective plaintext DTW preprocessing techniques. With two delicate treatments, such a strategy can quickly prune off unpromising time series sequences over the joint dataset, and avoid unnecessary computation for the complicated DTW distance. Specifically, the first one is that clustering can be performed by each hospital beforehand over their local dataset. Computation is firstly carried out between the query and the cluster centers so as to filter unpromising clusters (thus unpromising time series sequences) and produce a candidate set of clusters for further processing [3], [17]. The second treatment is based on the observation that instead of computing the DTW distance directly, a simpler distance could be computed so as to quickly exclude unpromising sequences that are not possible to be the best match [18]. Only promising sequences thus proceed to the complicated DTW computation.

With the above insight as a basis, we then consider how to achieve a secure and efficient realization of DTW-based medical analytics. Our main insight is to take advantage of lightweight cryptography (additive secret sharing) for efficient data encryption and properly work over the ciphertexts. To be compatible with the working paradigm of secret sharing and ease the healthcare institutions and querier from online participation, we delegate the secure analytics to two medical service providers which provide services collaboratively. Through careful examination on the sophisticated computation of DTW, we manage to decompose it into several atomic functions and propose their secure and efficient realizations using suitable secure computation techniques (including secret sharing and garbled circuits to cater for different network scenarios). With these secure customized functions, we design a complete protocol for the secure and scalable DTW-based analytics service.

We implement a system prototype in Java and perform an extensive evaluation over real-world medical time series sequences (ECG data) [19] to demonstrate the practical performance. Our result shows that a secure DTW-query computation over 15K ECG sequences, each of which consisting of 128 feature vectors, can be proceeded in 34 minutes.

The rest of this paper is organized as follows. Section 2 describes some preliminaries. Section 3 gives the system model and threat model. Section 4 presents the design overview. Section 5 details the secure distance functions. Section 6 elaborates on the proposed secure DTW-based medical analytic protocol. Section 7 formalizes the security definition and proof. The empirical evaluation is reported in Section 8, and the related work is investigated in Section 9. Section 10 concludes the whole paper.

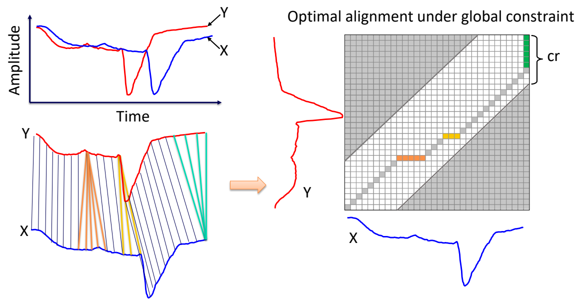

Fig. 1. The optimal alignment.

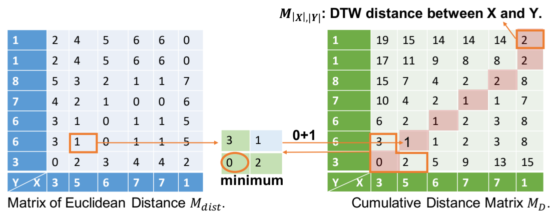

Fig. 2. An example of DTW distance between time series $X = (3, 5, 6, 7, 7, 1)$ and $Y = (3, 6, 6, 7, 8, 1, 1)$.

2 Background

2.1 Dynamic Time Warping

Dynamic time warping [18] (DTW) is a popular algorithm which can measure the similarity between time series data that may vary in time. Intuitively, as shown in Fig.1, the sequences are warped to match in a nonlinear manner, even if they shift along the time axis. Let $X = (\mathbf{x}_1, ..., \mathbf{x}_{|X|})$ denote a time series sequence of length $|X| \in \mathbb{N}$, and $Y = (\mathbf{y}_1, ..., \mathbf{y}_{|Y|})$ denote a time series sequence of length $|Y| \in \mathbb{N}$. These sequences consist of a series of dim-dimensional feature vectors $\mathbf{x}_i, \mathbf{y}_j$ for $i \in [1, |X|]$ and $j \in [1, |Y|]$. Note that two sequences are comparable only if their feature vectors have the same dimension. Further, we denote a subsequence of X as $X_i = (\mathbf{x}_1, ..., \mathbf{x}_i)$. Namely, X_i is a sequence starting from the first feature vector $\mathbf{x}_1$ till the i-th vector $\mathbf{x}_i$. Similarly, Y_j is a subsequence of Y from $\mathbf{y}_1$ to $\mathbf{y}_j$. Without loss of generality, we consider the sequences of the same length (i.e., $|X| = |Y|$) with integer data points. We further denote a dataset contains n-sequences as $\mathbf{Y} = \{Y^1, ..., Y^n\}$.

Given two time series sequences X and Y as input, the DTW algorithm outputs a distance-like quantity (called DTW distance) which measures the similarity of the two sequences. It is normally evaluated via a dynamic programming approach. The rationale is to find the optimal alignment with minimum cumulative distance between a slightly larger subsequences each time, until reaching the end of the sequences.

Algorithm 1 provides the details of the DTW algorithm. The $dist(\mathbf{x}_i, \mathbf{y}_j)$ is the underlying local distance, i.e., Euclidean Distance (ED), between all-pairwise feature vectors. The $D(X_i, Y_j)$ indicates the cumulative distance between subsequences X_i and Y_j. The minimum cumulative distance is denoted as D_{min} , and is declared as the local distance plus the minimum of its adjacent cumulative distances

Algorithm 1 The algorithm of DTW distance

```
Input: Time-series sequences $X = (\mathbf{x}_1, ..., \mathbf{x}_{|X|})$, $Y = (\mathbf{y}_1, ..., \mathbf{y}_{|Y|})$.
Output: The DTW distance $DTW(X, Y)$ between $X$ and $Y$.
1: for $i \in [1, |X|], j \in [1, |Y|]$ do
2:     $dist(\mathbf{x}_i, \mathbf{y}_j) \leftarrow (\mathbf{x}_i - \mathbf{y}_j)^2$.
3:     if $i == 1 \&\& j == 1$ then
4:         $D(X_1, Y_1) \leftarrow dist(\mathbf{x}_1, \mathbf{y}_1)$.
5:     else if $i == 1$ then
6:         $D(X_1, Y_j) \leftarrow dist(\mathbf{x}_1, \mathbf{y}_j) + D(X_1, Y_{j-1})$.
7:     else if $j == 1$ then
8:         $D(X_i, Y_1) \leftarrow dist(\mathbf{x}_i, \mathbf{y}_1) + D(X_{i-1}, Y_1)$.
9:     else
10:        $D_{min} \leftarrow min\{D(X_{i-1}, Y_{j-1}), D(X_{i-1}, Y_j), D(X_i, Y_{j-1})\}$.
11:        $D(X_i, Y_j) \leftarrow dist(\mathbf{x}_i, \mathbf{y}_j) + D_{min}$.
12:    end if
13: end for
14: $DTW(X, Y) \leftarrow D(X_{|X|}, Y_{|Y|})$
```

of the subsequences that one vector smaller than i, j. We formulate it as follows:

$$D(X_i, Y_j) = dist(\mathbf{x}_i, \mathbf{y}_j) + min\{D(X_{i-1}, Y_{j-1}), D(X_{i-1}, Y_j), D(X_i, Y_{j-1})\}. \tag{1}$$

The DTW distance between X and Y is denoted as $DTW(X, Y)$, and equivalent to $D(X_{|X|}, Y_{|Y|})$.

To facilitate understanding of the DTW algorithm, we provide in Fig. 2 a concrete example for demonstration. To calculate the DTW distance $DTW(X, Y)$, we first compute all pairwise local distances $dist(\mathbf{x}_i, \mathbf{y}_j)$ and record them in a matrix M_{dist}. Then, we iteratively compute cumulative distances $D(X_i, Y_j)$ according to Eq. (1) and fill in the cumulative distance matrix M_D. For example, when computing $D(X_2, Y_2)$ between subsequences $X_2(3, 5)$ and $Y_2(3, 6)$, we take the $dist(\mathbf{x}_2, \mathbf{y}_2) = dist(5, 6) = 1$ from the cell $M_{dist}(2, 2)$. Then we take its adjacent cumulative distances $D(X_1, Y_1), D(X_1, Y_2), D(X_2, Y_1)$. Afterwards, We calculate $D(X_2, Y_2) = dist(\mathbf{x}_2, \mathbf{y}_2) + min\{D(X_1, Y_1), D(X_1, Y_2), D(X_2, Y_1)\} = 1 + 0 = 1$, and record the result at the cell $M_D(2, 2)$. Eventually, the $DTW(X, Y)$ is located at the cell $M_D(6, 7)$ (i.e., the upright corner of the matrix M_D). The optimal alignment is the path consisting of all minimum cumulative distances, which are intermediate results, so the optimal path should be protected in our secure computation.

2.2 Cryptographic Preliminaries

Additive Secret Sharing: Additive secret sharing [10], [12] can be used as a highly efficient encryption scheme. On input an ℓ-bit value x, it generates two additive shares $\langle x \rangle_0^A, \langle x \rangle_1^A$ in the ring $\mathbb{Z}_{2^\ell}$ uniformly at random as ciphertexts, such that $\langle x \rangle_0^A + \langle x \rangle_1^A \equiv x \pmod{2^\ell}$. Each share $\langle x \rangle_i^A$ ($i \in \{0, 1\}$) can then be sent to a particular computing party denoted by P_i. Given two secret shares $\langle x \rangle^A$ and $\langle y \rangle^A$, secure addition and multiplication can be performed at the two parties P_0 and P_1 as follows. Unless an explicit claim, all operations are performed in $\mathbb{Z}_{2^\ell}$. For secure addition, each party P_i can simply compute $\langle x+y \rangle_i^A = \langle x \rangle_i^A + \langle y \rangle_i^A$ locally. For secure multiplication, the two parties need to have interaction which relies on the use of secret-shared *multiplication triple*. Suppose a multiplication triple (a_1, a_2, a_3) is secret-shared between the two parties, where a and b and random values, and $a_3 = a_1 \cdot a_2$, multiplication over secret-shared values proceeds as follows. Each party P_i first computes $\langle e \rangle_i^A = \langle x \rangle_i^A - \langle a_1 \rangle_i^A$ and $\langle f \rangle_i^A = \langle y \rangle_i^A - \langle a_2 \rangle_i^A$. Then, each party P_i broadcasts $\langle e \rangle_i^A$ and $\langle f \rangle_i^A$, and recovers e and f. Based on this, each party P_i can then compute $\langle x \cdot y \rangle_i^A = i \cdot e \times f + \langle a_1 \rangle_i^A \times f + \langle a_2 \rangle_i^A \times e + \langle a_3 \rangle_i^A$. Note that the multiplication triples independent of the data inputs can be generated in an offline phase, via some cryptographic protocols like correlated oblivious transfer (COT) [10], [20], or by an independent third party. Therefore, we assume that multiplication triples are pre-computed and available for online use in our design.

Boolean Sharing: The Boolean sharing [10], [13] can be viewed as the additive sharing over $\mathbb{Z}_2$. We denote the Boolean shares of a bit x as $[\![x]\!]_0, [\![x]\!]_1$. The addition operation in the additive sharing over $\mathbb{Z}_2$ is replaced by XOR operation ($\oplus$), and the multiplication operation is replaced by the AND operation ($\wedge$). In particular, similar to the multiplication, the AND operation is assisted by the precomputed Boolean AND Triple $[\![a_3]\!]_i = [\![a_1]\!]_i \wedge [\![a_2]\!]_i, i \in \{0, 1\}$.

Garbled Circuits: The garbled circuit (GC) protocol [11], [21] allows two parties say P_0 and P_1 holding inputs x_0 and x_1, respectively, to jointly evaluate an arbitrary function $\mathsf{f}(x_0, x_1)$. It ensures that only the function output is revealed in the end of the protocol, while the confidentiality of their inputs are protected against each other. Specifically, a party say P_0, called the *generator*, generates a garbled Boolean circuit computing $\mathsf{f}(\cdot, \cdot)$, a *mapping* between the garbled circuit outputs and the actual bits, and the *garbled label* $GI(x_0)$ corresponding to his input x_0. Then, P_0 sends the garbled circuit and $GI(x_0)$ to the other party P_1, called the *evaluator*. P_1 runs with P_0 a 1-out-of-2 Oblivious Transfer (OT_2^1 [22]) to obtain the garbled label $GI(x_1)$ associated with his secret input x_1 obliviously. With the garbled circuit and the garbled labels of inputs, P_1 evaluates the garbled circuit to obtain the garbled circuit output $GI(z)$, based on which P_1 can further learn the cleartext function output z according to the aforementioned mapping.

3 Problem Statement

3.1 Architecture

Fig. 3 provides an overview of the architecture of our secure DTW-based medical time series analytic system. At the core, there are three principals: the medical data owners (aka "*hospitals*" for brevity), the querier, and the medical analytics platform. The hospitals hold datasets of medical time series sequences and are willing to participate in medical services like disease screening for public health. In particular, the service allows the querier, holding a sample indicating a specific disease, to query over the hospitals' joint data to find similar medical time series data. In this paper, the widely popular DTW algorithm is adopted for the similarity measurement between two time series sequences. Due to privacy concerns, neither the hospitals nor querier would be willing to provide the cleartext medical time series data in the service. So, the time series data needs to be encrypted and the service has to run over the ciphertexts.

The multiple hospitals and the querier are bridged by the medical analytics platform, which facilitates the delivery of

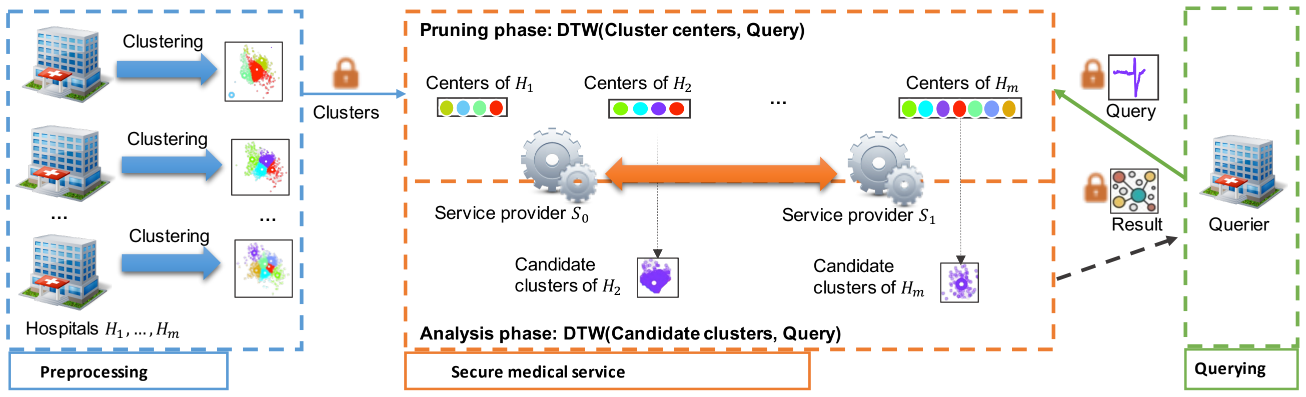

Fig. 3. System architecture.

the medical service in a secure and scalable manner. It collects encrypted medical time series data from the hospitals and the query sequence from the querier, and performs the disease screening. We consider that the platform is jointly run by two independent medical service providers (S_0 and S_1), who collaboratively deliver the secure service. Such a two-server model has seen an increasing use in different security applications with problem-specific customization (e.g., [15], [23], [24]). Our adoption follows the trend and newly explores the support for secure DTW-based medical services. In our system, the lightweight additive secret sharing technique, which is compatible with the two-serve model, is adopted for fast encryption of time series data on the hospitals and the querier. Each service provider then receives a share of the time series data and performs the secure DTW-based processing. Leveraging our observations from the advancements in cryptography (multi-party computation) and DTW (clustering and pruning), we develop a highly customized protocol to support the secure and scalable computation of DTW.

3.2 Threat Model

The threats in our system mainly come from the engagement of the medical analytic platform comprised of the two service providers. We assume them to be semi-honest and non-colluding parties. In particular, the two service providers will faithfully follow our protocol specification, yet they are interested in inferring the medical time series data of the hospitals and the querier and will do so independently. Note that such an assumption simply follows most of the prior works in the two-server model [15], [23], [24], [25]. We are aware that this also makes sense in practice as the service providers are business-driven parties and have little incentives to put their valuable reputation at risk due to behaving maliciously and colluding with each other [26], [27]. Additional rationale may include the existence of audits and the fear of legal or financial repercussions. The hospitals and the querier will honestly follow our protocol to properly supply the encrypted time series data to the two service providers for the secure medical analytic service.

In our system, we assume that each hospital establishes secure connection with the service providers to submit their data shares in parallel. It is noted that anonymity is fully orthogonal to our system and not the focus of our system which targets privacy protection of medical time series data in DTW-based analytics. Existing orthogonal solutions like mix-net [28] and Tor [29] could be readily adopted to hide the hospitals' IP addresses.

As mentioned before, our system may optionally resort to a third party dedicated to generating multiplication triples. Such a third party could be any computing service providers who are interested in providing the service. It is only for generating the triples and does not engage in the online protocol for secure DTW-based medical data analytics. The third party might be semi-honest, who will correctly generate the triples. Since the triples are data-independent and generated in an offline phase, no messages related with actual data for analytics would be accessible to a corrupted third party. When the third party is considered to be malicious, it may incorrectly generate the triples. In such case, the process of triple generation could be deployed to a trusted execution environment (TEE) [30] enabled on the third party side, like Intel SGX and ARM TrustZone, which can guarantee the computation integrity.

4 Design Overview

At a high level, our protocol for the secure DTW-based medical service is comprised of three phases: Setup, Pruning, and Analysis. In the Setup phase, each hospital and the querier pre-processes their medical time series data so as to boost the scalability of the secure DTW-based medical service, and they then perform data encryption under additive secret sharing, from which the produced shares are sent to the service providers respectively. Specifically, in our protocol we make use of observations from the plaintext domain [17], [18] and propose a hybrid pruning strategy to be used in the Pruning phase and Analysis, which requires some pre-processing on the local side.

The hybrid pruning strategy is comprised of two delicate treatments. The first treatment is based on the insight that we can let each hospital do clustering over their time series dataset and produces a set of clusters along with their cluster centers. To match a query against the data from the hospitals, computation can be first carried out between the query and the cluster centers to filter unpromising clusters (thus unpromising time series sequences) and produce a candidate set of clusters for further processing. We note that such clustering over the time series data on each hospital

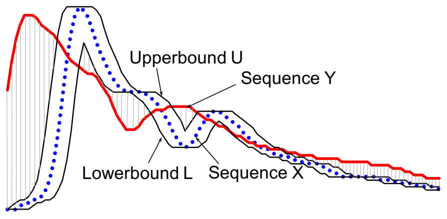

Fig. 4. Illustration of the lower bounding distance.

can be well supported by an algorithm named density peak clustering [3], [17], which has been shown to be effective for use in computing DTW over time-series datasets with arbitrary shapes (unlike R-tree based DBSCAN, and k-means and its variants).

The second treatment is based on the observation that before computing the relatively complex DTW distance between two time series sequences, a simpler distance named lower bounding distance could be computed so as to quickly exclude unpromising time series sequences that are not possible to be the best match [18]. In particular, only those time series sequences whose lower bounding distances with the query time series sequence are within a certain threshold proceed to the computation of the DTW distance. Note that this is also applicable for doing comparison between the query times series sequence and cluster centers. To support such treatment, the querier needs to do some pre-processing on the query sequence and produce the bounding values for use in computing the lower bounding distances.

With these two delicate treatments realized properly in a secure manner, the Pruning phase in our protocol would produce a candidate set of clusters of encrypted sequences. In the Analysis phase, the matched (encrypted) sequences among the candidates are finally located, realizing the secure medical service for disease screening. To realize these two phases securely, our main insight is to decompose the computation into atomic functions and build on the secure computation techniques (additive secret sharing and garbled circuits) to appropriately realize secure versions of these atomic operations. In particular, we identify the needs for the following secure functions in order to build the secure DTW-based medical analytic protocol: Secure Squared Euclidean Distance Function (SSED), Secure Lower Bounding Distance Function (SLB), and Secure DTW Distance Function (SDTW). We detail the design of these secure functions in Section 5 and give the complete protocol for secure DTW-based medical analytics in Section 6.

5 Secure Distance Functions

5.1 Secure Squared Euclidean Distance Function

Suppose that S_0 and S_1 already obtain the shared vectors of query X and the candidate sequence Y, and the shares of their squared values, denoted as $\langle \mathbf{x} \rangle^A$ and $\langle \mathbf{y} \rangle^A$, and $\langle \mathbf{x}^2 \rangle^A$ and $\langle \mathbf{y}^2 \rangle^A$. They also have a bunch of pre-generated MTs. S_0 and S_1 run the function SSED to attain $\langle dist \rangle_b^A = \langle \mathbf{x}^2 \rangle_b^A + \langle \mathbf{y}^2 \rangle_b^A - 2(\mathsf{Mul}^A(\langle \mathbf{x} \rangle_b^A, \langle \mathbf{y} \rangle_b^A))$, where $b \in \{0, 1\}$.

5.2 Secure Lower Bounding Distance Function

5.2.1 *Overview*

As mentioned before, our design makes an observation in the literature on DTW that a lower bounding distance can be leveraged to quickly prune off sequences that are not possible to be the best match ahead of DTW computation. We resort to the widely adopted algorithm of Keogh et al. [18] to compute the lower bounding distance. As illustrated in Fig. 4, given a query X, the algorithm defines an upperbound U and a lowerbound L surrounding it. Given a constant number cr, U consists of a set of vectors $\mathbf{u}_i = max(\mathbf{x}_{i-cr} : \mathbf{x}_{i+cr})$, and L consists of $\mathbf{l}_i = min(\mathbf{x}_{i-cr} : \mathbf{x}_{i+cr})$ that $\forall_i, \|\mathbf{u}_i\| \geq \|\mathbf{x}_i\| \geq \|\mathbf{l}_i\|$. Given a candidate Y, the lower bounding distance LB is formulated as follows:

$$LB(X,Y) = \sqrt{\sum_{i=1}^{|X|} \begin{cases} (\mathbf{y}_i - \mathbf{u}_i)^2 & \text{if } \|\mathbf{y}_i\| > \|\mathbf{u}_i\| \\ (\mathbf{y}_i - \mathbf{l}_i)^2 & \text{if } \|\mathbf{y}_i\| < \|\mathbf{l}_i\| \\ 0 & \text{otherwise} \end{cases}} \quad (2)$$

It is provably tight $LB(X,Y) \leq DTW(X,Y)$ and holds linear time complexity. As the function of Euclidean distance is monotonic and concave, we omit the square root operation for the ease of integration with secure computation.

Through careful examination on the LB function, we note that its computation in the ciphertext domain could be formulated in this way. Let uy and yl denote the pairwise EDs between U (or L) and the sequence Y. The computation can be formulated as: **if** $(\|\mathbf{y}_i\| > \|\mathbf{u}_i\|)$ **then** $dist_1 \leftarrow uy$ **else** $dist_1 \leftarrow 0$; and **if** $(\|\mathbf{l}_i\| > \|\mathbf{y}_i\|)$ **then** $dist_2 \leftarrow yl$ **else** $dist_2 \leftarrow 0$. And the distance is just the sum of $dist_1$ and $dist_2$. Given the implication that the value of $\mathbf{l}_i$ is always smaller than the value of $\mathbf{u}_i$, such formulation correctly deduces three conditions $(\|\mathbf{y}_i\| > \|\mathbf{u}_i\| \&\& \|\mathbf{y}_i\| > \|\mathbf{l}_i\|)$, $(\|\mathbf{y}_i\| < \|\mathbf{u}_i\| \&\& \|\mathbf{y}_i\| > \|\mathbf{l}_i\|)$, and $(\|\mathbf{y}_i\| < \|\mathbf{u}_i\| \&\& \|\mathbf{y}_i\| < \|\mathbf{l}_i\|)$ corresponding to the result that the lower bounding distance could be uy, 0, or yl, respectively. So, such formulation largely simplifies the computation as two branching functions running in parallel, facilitating the computation in encrypted domain. With such formulation in mind, we then investigate the secure realization of the lower bounding distance function SLB. Below we give the construction of two essential cryptographic gadgets: the secure comparison gadget and the secure branching gadget, and show how to build SLB from them.

5.2.2 *Secure Comparison Gadget*

The secure comparison gadget (SCMP) takes as input the secret shares of two values, determines their rank, and outputs a bit indicating the comparison result. To support secure comparison, we note that a common approach is to use garbled circuits. This can work in our scenario in that one service provide can play the role of the generator while the other can play the role of the evaluator. At a high level, we can build on a Boolean circuit which takes as input the secret sharings of the two values, reconstructs the two values, does the comparison, and outputs the comparison result. However, such garbled circuit-based approach does incur intensive resource demands in both computation and communication. It is better suited for the scenario where the protocol has to run in high-latency network environments due to the property of constant communication round.

In this work, in addition to the common garbled circuit approach, we also investigate an alternative approach that works in the secret sharing domain and achieves high efficiency in both computation and communication. We make an observation from the recent work [16] that the secure comparison problem can be transformed into a simpler bit extraction problem that can be realized in the secret sharing domain. The idea is introduced as follows. Let z denote the subtraction result between two values a_1 and a_2, i.e., $z = a_1 - a_2$. Note that the secure subtraction can be easily done in the secret sharing domain, i.e., $\langle z\rangle^A \in \mathbb{Z}_{2^\ell} = \langle a_2\rangle^A - \langle a_1\rangle^A$. Given a sufficiently large ℓ, the values in the ring $\mathbb{Z}_{2^\ell}$ would be separated into two halves: the lower half ($[0, 2^{\ell-1} - 1]$) for non-negative values, and the upper half ($[2^{\ell-1}, 2^\ell - 1]$) for negative values. The most-significant bit (MSB) thus would be 0 for a non-negative value z, and otherwise 1 for a negative value z. As long as the MSB bit z_ℓ of z is known, the comparison result between a_1 and a_2 can be obtained. So, the need here for secure comparison is to extract the MSB of the subtraction result z in the secret sharing domain. To cater for this need, we employ the bit extraction protocol in [16], which is able to extract the secret sharing of the MSB of the subtraction result z, given the secret sharings of two values a_1 and a_2. As will be shown in our experiments, with such new way of doing secure comparison as an atomic operation in our whole protocol for secure DTW-based medical analytics, the total online runtime in the LAN setting is $10\times$ faster than the case when garbled circuit is used. It is noted that as the bit extraction-based approach inherits a linear round complexity, it is well suited for the scenario that a dedicate network is established in between the two non-colluding service providers.

Based on what has been introduced as far, we now elaborate on how to obtain realization of the secure comparison gadget based on garbled circuits and secret sharing respectively.

Realization based on Garbled Circuits: Given the service S_1 as the *generator* and the service S_0 as the *evaluator*, the $\mathsf{SCMP}_{gc}(\langle a_1\rangle^A, \langle a_2\rangle^A)$ gadget proceeds as follows:

1) S_1 sends to S_0 a pre-built circuit of the comparison function f, and the garbled labels of his shares $GI(\langle a_1\rangle_1^A), GI(\langle a_2\rangle_1^A)$. The function f switches $\langle a_1\rangle^A$ and $\langle a_2\rangle^A$ to a_1 and a_2 via modular additions, and obtain the comparison bit z, where $z = 0$ if $a < b$ and $z = 1$ if $a_1 \geq a_2$.
2) If the rank between a and b is required to be hidden, S_1 generates the randomness $r \in \mathbb{Z}_{2^\ell}$, and sets the output of the circuits as $(z - r) \bmod 2^\ell$.

Realization based on Secret Sharing: Given the two service providers, the pre-generated Multiplication triples and the Boolean AND triples, the gadget $\mathsf{SCMP}_{ss}(\langle a_1\rangle^A, \langle a_2\rangle^A)$ proceeds as the follows:

1) S_b initializes the variables for the additive shares $\langle t_1\rangle_b^A, \langle t_2\rangle_b^A \in \mathbb{Z}_{2^\ell}$, and the variables for the Boolean shares $[\![w_k]\!]_b, [\![p_k]\!]_b, [\![q_k]\!]_b, [\![c_k]\!]_b, [\![d_k]\!]_b, [\![e_k]\!]_b \in \mathbb{Z}_2$, where $k \in [1, \ell]$ and $b \in \{0, 1\}$.
2) S_b computes $\langle z\rangle_b^A = \langle a_1\rangle_b^A - \langle a_2\rangle_b^A \bmod 2^\ell$.
3) S_b decomposes $\langle z\rangle_b^A$ as the bit string $\langle z_k\rangle_b^A$, where $k \in [1, l]$. S_0 then sets $[\![w_k]\!]_0 = \langle z_k\rangle_0^A$, $[\![p_k]\!]_0 = \langle z_k\rangle_0^A$, and $[\![q_k]\!]_0 = 0$. Meanwhile, S_1 sets $[\![w_k]\!]_1 = \langle z_k\rangle_1^A$, $[\![p_k]\!]_1 = 0$, and $[\![q_k]\!]_1 = \langle z_k\rangle_1^A$.
4) S_0 and S_1 compute the following steps to extract the Most Significant Bit (MSB) of the $\langle z\rangle_b^A$:
   a) S_0 and S_1 compute $[\![d_k]\!]_b = [\![p_k]\!] \cdot [\![q_k]\!]$ in a batch to reduce the number of rounds of interaction, and obtain their shares $[\![d_k]\!]_b$, respectively;
   b) S_b sets $[\![c_1]\!]_b = [\![d_1]\!]_b$;
   c) For $k \in [2, l - 1]$, S_b computes $[\![d_k]\!]_b = [\![d_k]\!]_b + b$ at local; S_0 and S_1 jointly compute $[\![e_k]\!]_b = [\![w_k]\!] \cdot [\![c_{k-1}]\!] + b$, and $[\![c_k]\!]_b = [\![e_k]\!] \cdot [\![d_k]\!] + b$ with the assist of the pre-generated Boolean AND Triples;
   d) Then, S_b computes to obtain his share of the comparison bit $[\![z_l]\!]_b = [\![w_l]\!]_b + [\![c_{l-1}]\!]_b$.
5) If the rank is required to be viewed in plaintext value, then S_0 and S_1 recover z_l.
6) To convert the comparison bit from over $\mathbb{Z}_2$ to $\mathbb{Z}_{2^\ell}$, S_0 sets $\langle t_1\rangle_0 = [\![z_l]\!]_0$, and $\langle t_2\rangle_0^A = 0$, while S_1 sets $\langle t_1\rangle_1 = 0$, and $\langle t_2\rangle_1^A = [\![z_l]\!]_1$. At the end, S_0 and S_1 jointly compute $\langle z_l\rangle_b^A = \langle t_1\rangle_b^A + \langle t_2\rangle_b^A - 2 \cdot \langle t_1\rangle^A \cdot \langle t_2\rangle^A$ to obtain their shares of comparison bit, respectively.

Remark: To further reduce the latency due to the interactions for multiplications in the above secret sharing-based protocol, our implementation delicately carries out a part of the computation in a batch, and manages to reduce the round complexity from 3ℓ to 2ℓ. Such acceleration is not trivial when considering our secure computation, as the data volume is large, and the ℓ is demanded to be large enough.

5.2.3 Secure Branching Gadget

For each feature vector of the candidate sequence as $\mathbf{y}$, the upperbound $\mathbf{u}$, and the lowerbound as $\mathbf{l}$, given the pairwise secure squared Euclidean distances uy, yl between them, a secure branching gadget ($\mathsf{SBranch}$) needs to be conducted. The functionality of the $\mathsf{SBranch}$ gadget is to choose one of uy, yl, or 0 based on the rank of the variables $\mathbf{y}$, $\mathbf{u}$, $\mathbf{l}$. A vanilla solution is to compare $\mathbf{y}$ and $\mathbf{u}$ (or $\mathbf{l}$) and output a comparison bit indicating their rank, so that the service providers can select the result from $dist_1$ and $dist_2$. Yet, revealing the rank of every feature vector in Y and $\{U, L\}$ endows an adversary the ability to deduce the range of query X, and tighten the estimation via adaptive testings with a series of evenly incremented values of Y. Thus, it is desired to devise an oblivious branching scheme, which solves two branching functions in parallel, and assigns the shares of distance to S_0 and S_1 without leaking the rank. We give two secure realizations based on garbled circuits and secret sharing respectively.

Realization based on Garbled Circuits: Given the service S_1 as the *generator* and the service S_0 as the *evaluator*. Let $\mathbf{m_1}, \mathbf{m_2}$ denote the conditions, and c_1, c_2 denote the decisions. The $\mathsf{SBranch}_{gc}(\langle \mathbf{m_1}\rangle^A, \langle \mathbf{m_2}\rangle^A, \langle c_1\rangle^A, \langle c_2\rangle^A)$ proceeds as the follows:

1) S_0 generates $\omega \in_R \mathbb{Z}_{2^\ell}$.
2) S_1 sends to S_0 a pre-built circuit of the above introduced branching function f, and $GI(\langle \mathbf{m_1}\rangle_1^A), GI(\langle \mathbf{m_2}\rangle_1^A),\ GI(\langle c_1\rangle_1^A), GI(\langle c_2\rangle_1^A)$. The function f performs the following steps:

Algorithm 2 Secure Lower Bounding Distance Function

```
1: function SLB($\langle U\rangle^A, \langle U^2\rangle^A, \langle L\rangle^A, \langle L^2\rangle^A, \langle Y\rangle^A, \langle Y^2\rangle^A$)
2:     $S_b$ initializes $\langle LB\rangle_b^A$ for $b \in \{0,1\}$.
3:     for $i \in [1, |X|]$ do
4:         $S_b$ initializes $\langle dist_1\rangle_b^A$, $\langle dist_2\rangle_b^A$, $\langle uy\rangle_b^A$ and $\langle yl\rangle_b^A$.
5:         $S_0$ and $S_1$ run to get $\langle uy\rangle_b^A \leftarrow$ SSED($\langle \mathbf{u}_i\rangle^A, \langle \mathbf{y}_i\rangle^A, \langle \mathbf{y}_i^2\rangle^A, \langle \mathbf{u}_i^2\rangle^A$ ).
6:         $S_0$ and $S_1$ run to get $\langle yl\rangle_b^A \leftarrow$ SSED($\langle \mathbf{l}_i\rangle^A, \langle \mathbf{y}_i\rangle^A, \langle \mathbf{y}_i^2\rangle^A, \langle \mathbf{l}_i^2\rangle^A$ ).
7:         $S_0$ and $S_1$ run to get $\langle dist_1\rangle_b^A \leftarrow$ SBranch($\langle \mathbf{u}_i\rangle^A, \langle \mathbf{y}_i\rangle^A, \langle uy\rangle^A, 0$).
8:         $S_0$ and $S_1$ run to get $\langle dist_2\rangle_b^A \leftarrow$ SBranch($\langle \mathbf{y}_i\rangle^A, \langle \mathbf{l}_i\rangle^A, \langle yl\rangle^A, 0$).
9:         $S_b$ sets $\langle dist\rangle_b^A = \langle dist_1\rangle_b^A + \langle dist_2\rangle_b^A$, and $\langle LB\rangle_b^A = \langle LB\rangle_b^A + \langle dist\rangle_b^A$.
10:    end for
11:    $S_b$ gets $\langle LB\rangle_b^A$.
12: end function
```

a) switches $\langle \mathbf{m_1}\rangle^A$, $\langle \mathbf{m_2}\rangle^A$, $\langle c_1\rangle^A$, $\langle c_2\rangle^A$ to $\mathbf{m_1}$, $\mathbf{m_2}$, c_1, c_2 via modular additions;
b) compares $\mathbf{m_1}$ and $\mathbf{m_2}$, and sets a bit b that $b = 0$ if $\|\mathbf{m_1}\| < \|\mathbf{m_2}\|$ and $b = 1$ if $\|\mathbf{m_1}\| \geq \|\mathbf{m_2}\|$;
c) sets the result z, where if $b = 0$, $z = (c_1 - \omega) \bmod 2^\ell$; and if $b = 1$, $z = \langle dist\rangle_0^A = (c_2 - \omega) \bmod 2^\ell$.

3) After S_0 obtains the garbled labels of his shares via OT, S_0 evaluates the circuit to get $GI(z)$.
4) S_0 decodes $GI(z)$ to obtain his share of the branching decision as $\langle dist\rangle_0^A = z \bmod 2^\ell$, and S_1 sets his share as $\langle dist\rangle_1^A = \omega$.

Realization based on Secret Sharing: Given the two service providers, the pre-generated Beaver's Triples, and the SCMP$_{ss}$ gadget, the SBranch$_{ss}$($\langle \mathbf{m_1}\rangle^A$, $\langle \mathbf{m_2}\rangle^A$, $\langle c_1\rangle^A$, $\langle c_2\rangle^A$) proceeds as the follows:

1) S_0 and S_1 jointly execute the SCMP$_{ss}$ gadget $\langle z\rangle_b^A \leftarrow$ SCMP$_{ss}$($\langle \mathbf{m}_1\rangle^A, \langle \mathbf{m}_2\rangle^A$) to compare the conditions $\mathbf{m_1}, \mathbf{m_2}$, and obtain their shares of the comparison bit $\langle z\rangle_b^A$, where $\langle z\rangle_b^A \in \mathbb{Z}_{2^\ell}$ and $b \in \{0,1\}$.
2) Based on $\langle z\rangle_b^A$, one of the decisions c_1, c_2 is assigned as the resulting distance. Both service providers calculate their shares of the distance obliviously.
3) That is, S_b computes $\langle p\rangle_b^A = b - \langle z\rangle_b^A$.
4) At the end, S_0 and S_1 jointly compute $\langle dist\rangle_b^A = \langle p\rangle^A \cdot \langle c_1\rangle^A + \langle z\rangle^A \cdot \langle c_2\rangle^A$.

5.2.4 *Secure Lower Bounding Distance Algorithm*

Given the cryptographic gadgets SCMP and SBranch, the SLB function is described in Algorithm 12. On input a set of shares of all feature vectors in Y, U, and L, and the shares of their squared values, the SLB function outputs the secure LB distance between query and candidate sequence. It invokes the SSED function to securely calculate the squared Euclidean distance between feature vectors $\mathbf{u}_i, \mathbf{y}_i$ and $\mathbf{l}_i, \mathbf{y}_i$. And it invokes the SBranch gadget to choose correct and well-masked distance $dist_1$ (or $dist_2$) based on the rank of feature vectors. Here the SBranch gadget can be instantiated by the previously introduced two realizations, i.e., SBranch$_{gc}$ and SBranch$_{ss}$. Finally, it sums all obtained $dist_1$ and $dist_2$ as the result LB.

5.3 Secure DTW Distance Function

The secure DTW distance function (SDTW) is the core component in our secure DTW-based analytic system. It utilizes the dynamic programming approach to produce the minimum cumulative distance D_{min} between the query X and each candidate sequence Y in an iterative fashion. To assist the find minimum calculation, we now propose the secure find minimum gadget.

5.3.1 *Secure Find Minimum Gadget*

The secure find minimum gadget (SFindMin) takes as input three given shares, determines the minimum value among them, and generates new shares of the minimum value as the output. With the re-generation of shares, the index of the minimum cumulative distance D_{min} in every iteration is hidden. This operation cannot be neglected. Otherwise, the optimal warping path of X and Y will be revealed, and further allows the adversary to estimate X or Y, depending on which party is compromised.

Realization based on Garbled Circuits: Given S_1 as the *generator* and S_0 as the *evaluator*, the SFindMin$_{gc}$($\langle a_1\rangle^A, \langle a_2\rangle^A, \langle a_3\rangle^A$) gadget proceeds as the follows:

1) S_1 generates the randomness $r \in \mathbb{Z}_{2^\ell}$. Then, S_1 sends to S_0 a pre-built circuit of the function f, and $GI(\langle a_1\rangle_1^A), GI(\langle a_2\rangle_1^A), GI(\langle a_3\rangle_1^A)$ and $GI(r)$. The function f performs the following steps:
   a) switches $\langle a_1\rangle^A$, $\langle a_2\rangle^A$ and $\langle a_3\rangle^A$ to a_1, a_2 and a_3 via modular additions;
   b) compares a_1, a_2 and a_3 to get the minimum as D_{min};
   c) generates new shares of D_{min} via modular subtraction, i.e., $z = (D_{min} - r) \bmod 2^\ell$.
2) S_0 obtains $GI(\langle a_1\rangle_0^A), GI(\langle a_2\rangle_0^A), GI(\langle a_3\rangle_0^A)$ via OT, and evaluates the circuit to obtain $GI(z)$.
3) At the end, S_0 decodes $GI(z)$ and sets his share as $\langle D_{min}\rangle_0^A = z$. Accordingly, S_1 sets his share as $\langle D_{min}\rangle_1^A = r$.

Realization based on Secret Sharing: Given the two service providers, the SBranch$_{ss}$ gadget, the SFindMin$_{ss}$($\langle a_1\rangle^A, \langle a_2\rangle^A, \langle a_3\rangle^A$) gadget proceeds the following steps:

1) S_0 and S_1 jointly compute $\langle D_1\rangle_b^A \leftarrow$ SBranch$_{ss}$($\langle a_1\rangle^A, \langle a_2\rangle^A, \langle a_1\rangle^A, \langle a_2\rangle^A$) to get a smaller one of $\langle a_1\rangle^A, \langle a_2\rangle^A$, where $b \in \{0,1\}$.
2) Afterwards, S_0 and S_1 jointly compute $\langle D_2\rangle_b^A \leftarrow$ SBranch($\langle D_1\rangle^A, \langle c\rangle^A, \langle D_1\rangle^A, \langle a_3\rangle^A$) to get a smaller one from $\langle c\rangle^A$ and the previous result.
3) Finally, S_b set his share of the minimum value as $\langle D_{min}\rangle_b^A = \langle D_2\rangle_b^A$, respectively.

5.3.2 *Secure DTW Distance Algorithm*

Given the SFindMin gadget, Algorithm 3 describes the SDTW function. It takes as input the shares of all feature vectors and their squared values of the candidate sequence Y and the query X, and outputs to the service providers the shares of the DTW distance $\langle DTW\rangle^A$ between them. A common practice for DTW is to apply a global constraint [31] to avoid pathological warping [18], where a relatively small portion of one sequence would not be warped to map a

Algorithm 3 Secure DTW Function

```
1: function SDTW($\langle X\rangle^A$, $\langle X^2\rangle^A$, $\langle Y\rangle^A$, $\langle Y^2\rangle^A$)
2:     $S_b$ initializes $\langle DTW\rangle_b^A$ and two arrays $cost_b$ and $cost_{prev,b}$ with numeric infinity (INF), that each has $(2cr+1)$ numbers of cells for $b \in \{0,1\}$.
3:     for $i \in [0, |X|-1]$ do
4:         $S_b$ initializes $k \leftarrow max(0, cr-i)$.
5:         for $j \in [\ \max(0, i-cr), \min(|X|-1, i+cr)]$ do
6:             $S_b$ initializes $\langle a_1\rangle_b^A$, $\langle a_2\rangle_b^A$, $\langle a_3\rangle_b^A$ with INF, $\langle dist\rangle_b^A$, and $\langle D\rangle_b^A$.
7:             If $i=0$ && $j=0$, $S_0$ and $S_1$ run to get $\langle dist\rangle_b^A \leftarrow$ SSED($\langle \mathbf{x}_0\rangle^A$, $\langle \mathbf{y}_0\rangle^A$, $\langle \mathbf{x}_0^2\rangle^A$, $\langle \mathbf{y}_0^2\rangle^A$). Then, $S_b$ stores his share at $cost_b[k]$. $S_b$ sets $k$ ++, and continues to next iteration.
8:             If $j \geq 1$ && $k \geq 1$, $S_b$ sets $\langle a_2\rangle_b^A \leftarrow cost_b[k-1]$; else INF.
9:             If $i \geq 1$ && $k+1 \leq 2*cr$, $S_b$ sets $\langle a_1\rangle_b^A \leftarrow cost_{perv,b}[k+1]$; else INF.
10:            If $i \geq 1$ && $j \geq 1$, $S_b$ sets $\langle a_3\rangle_b^A \leftarrow cost_{prev,b}[k]$; else INF.
11:            $S_0$ and $S_1$ run to get $\langle D\rangle_b^A \leftarrow$ SSED($\langle \mathbf{x}_i\rangle^A$, $\langle \mathbf{y}_j\rangle^A$, $\langle \mathbf{x}_i^2\rangle^A$, $\langle \mathbf{y}_j^2\rangle^A$) + SFindMin ($\langle a_1\rangle^A$, $\langle a_2\rangle^A$, $\langle a_3\rangle^A$). Then, $S_b$ stores his share at $cost_b[k]$, and then sets $k$ ++.
12:        end for
13:        $S_b$ copies $cost_b$ to $cost_{prev,b}$, and cleans $cost_b$.
14:    end for
15:    $S_b$ gets $\langle DTW\rangle_b^A \leftarrow cost_{prev,b}[cr]$.
16: end function
```

considerably large portion of another. It introduces a cr-width sliding window so that only the elements within the window can be compared. We follow the common practice, and securely realize the global constraint in our SDTW function. As suggested [18], [32], the parameter cr is a constant and independent on the data, we thus assume it is a hyper parameter known by both service providers.

We describe the way to optimize the memory consumption when calculating DTW distance. Recall, DTW between X and Y is normally realized by maintaining an $|X|$-by-$|X|$ matrix M_D filled with cumulative distances $D(X_i, Y_j)$ for $i, j \in [1, |X|]$. We observe that the calculation of $D(X_i, Y_j)$ is only related to its adjacent cumulative distances, i.e., $D(X_{i-1}, Y_{j-1}), D(X_{i-1}, Y_j), D(X_i, Y_{j-1})$. Thereby, rather than maintaining the whole matrix in memory, recording the $D(X_i, \{Y_j\}_1^{|Y|})$ and $D(X_{i-1}, \{Y_j\}_1^{|Y|})$ is enough. To do so, we can use two $2cr+1$-length arrays $cost$ and $cost_{prev}$ to store the i-th column $\mathbf{m}_{i,*}$ (i.e. current column) and $(i-1)$-th column $\mathbf{m}_{i-1,*}$ (i.e. previous column) of M_D. For the entire SDTW calculation, they act as two vertical bars moving from left to right and bottom-up one cell each iteration. Thus, the quadratic storage consumption can be saved to linear with the length of sequence. With this observation in mind, we briefly sketch how to realize the SDTW function.

For simplicity of representation and without loss of generality, we take the calculation of $D(X_i, Y_j)$ as an example. Assume that $cost_{prev}$ is already filled with $D(X_{i-1}, \{Y_j\}_{i-cr-1}^{i+cr-1})$, i.e., a replica of the $cost$ in previous iteration. And the $cost$ is awaiting to be filled with $D(X_i, \{Y_j\}_{i-cr}^{i+cr})$ in the current iteration. Suppose that cell $cost[k]$ stores $D(X_i, Y_j)$, where $k \in [max(0, cr-i), 2cr+1]$. When filling it, the SDTW function invokes the SSED function to calculate $dist(\mathbf{x}_i, \mathbf{y}_j)$. Afterwards, we try to obtain D_{min} among its adjacent cumulative distances that are already stored at $cost[k-1]$, $cost_{prev}[k+1]$ and $cost_{prev}[k]$. Precedently, we need to examine if the awaiting computed $D(X_i, Y_j)$ is located at the edge of the sliding window, i.e., at $cost_b[0]$ or $cost_b[2cr+1]$. If not, the SDTW function invokes the SFindMin gadget to determine D_{min}. Note that, the SFindMin gadget here can be realized by the aforementioned SFindMin$_{gc}$ or SFindMin$_{ss}$ to suit different runtime environments. Otherwise, it compares the existences of the three adjacent cumulative distances to find the minimum. At the end, we add $dist(\mathbf{x}_i, \mathbf{y}_j)$ and D_{min} together as $D(X_i, Y_j)$, and store into cell $cost[k]$. The function calculates the above procedures repeatedly until reaching $X_{|X|}, Y_{|Y|}$. Then, $D(X_{|X|}, Y_{|Y|})$ is the DTW distance between X and Y. Following the above methodology, we provide the details of secure realization of the SDTW function in Algorithm 3.

6 Protocol for Secure DTW-based Medical Analytics

Fig. 5 is a high-level illustration of the online phases, i.e., the Pruning phase and the analytic phase. It shows the workflow in each phase, and the functionality dependence between the distance functions and the cryptographic gadgets they rely on. In the following section, we present the details of the setup phase, and the two online phases, and summarize the whole protocol Φ in Fig. 6.

6.1 Setup Phase

During the Setup phase, each party prepares the data locally that can be used in online execution. Suppose that m hospitals $\{H_1, ..., H_m\}$ join the computation. Each hospital H_α holds a dataset $\mathbf{Y}^\alpha$ consisting of n sequences $\{Y^1, ..., Y^n\}$, where $\alpha \in [1, m]$. During preprocessing, each hospital prepartitions the local dataset into k clusters $\mathbf{C}_1 = \{C^1, ..., C^k\}$ based on the Density Peak clustering [17] as mentioned before. And the clusters are represented by the cluster centers $\{\mathfrak{C}^1, ..., \mathfrak{C}^k\}$. Given the pre-generated upperbound U, lowerbound L, and the matching threshold Δ held by the querier Q. We assume X and Y are equal length, i.e., both consist of $|X|$ numbers of dim-dimensional vectors.

Once the Setup phase is launched, the querier Q and all hospitals produce additive shares of vectors and their squared values of all the sequences they hold, and deploy the corresponding shares to the service providers S_0 and S_1 for coordinate processing. To facilitate the multiplications on secret shares, S_0 and S_1 interactively produce a bunch of dim-dimensional MTs $(\langle \mathbf{a_3}\rangle^A, \langle \mathbf{a_1}\rangle^A, \langle \mathbf{a_2}\rangle^A)$ with the relationship $\mathbf{a_3} = \mathbf{a_1} \times \mathbf{a_2}$. If the proposed gadgets (SBranch, SFindMin and SCMP) are realized based on Secret Sharing, both service providers produce the Boolean AND triples. Note that one triple should be used only for a single multiplication or Boolean AND operation, and be discarded once it is used. If we adopt the GC-based realizations, S_1 as the generator, generates garbled circuits for the three gadgets.

6.2 Secure Pruning Phase

The Pruning phase is used to eliminate the unpromising clusters whose centers are not similar to the query, thus to reduce the volume of data submitted to the sequential comparison in the Analysis phase. To execute in parallel, S_0 and S_1 invoke m instances, respectively. A pair of instances

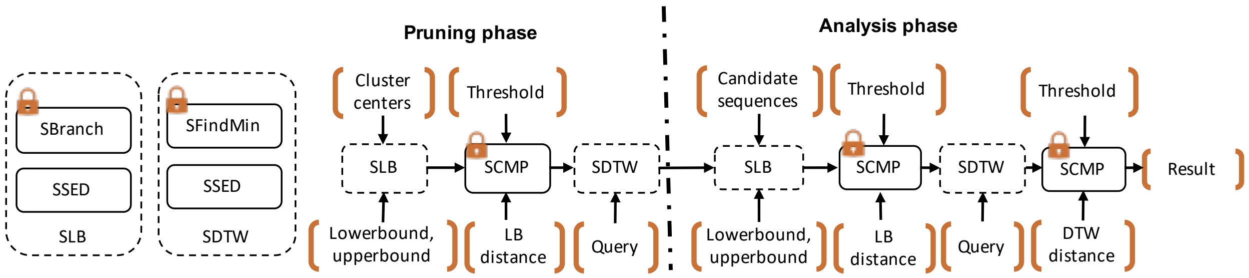

Fig. 5. Online phases overview.

```
Protocol Φ:
Setup($X, U, L, \{\mathbf{C}_1, ..., \mathbf{C}_m\}, \Delta$):
  1: $S_0$ and $S_1$ interactively generate $M$ sets of $MTs$.
  2: $Q$ locally generates the shares $\langle\Delta\rangle^A$, $\langle X\rangle^A$, $\langle X^2\rangle^A$, $\langle U\rangle^A$, $\langle U^2\rangle^A$, $\langle L\rangle^A$, $\langle L^2\rangle^A$, and distributes them to $S_0$ and $S_1$.
  3: Each $H_\alpha$ locally generates the shares $\langle Y\rangle^A$ and $\langle Y^2\rangle^A$ of $Y \in \mathbf{Y}_\alpha$, $\langle\mathfrak{C}^\beta\rangle^A$ and $\langle(\mathfrak{C}^\beta)^2\rangle^A$ of $\mathfrak{C}^\beta$, and dummy identifiers
     $r_\beta \in \{r_1, ..., r_k\}$ randomly for each cluster, where $\beta \in [1, k]$. Then, each $H_\alpha$ independently distributes all shares to $S_0$
     and $S_1$ for coordinate processing.
Pruning($\langle U\rangle^A, \langle U^2\rangle^A, \langle L\rangle^A, \langle L^2\rangle^A, \langle X\rangle^A, \langle X^2\rangle^A, \langle\mathfrak{C}^\beta\rangle^A, \langle(\mathfrak{C}^\beta)^2\rangle^A, \langle\Delta\rangle^A$):
  1: $S_b$ initializes his array $\mathbf{Arr}^C_{\alpha,b}$,where $b \in \{0, 1\}$.
  2: for each cluster center $\mathfrak{C}^\beta \in \mathbf{C}_\alpha$ in each $H_\alpha$ do
  3:     $S_0$ and $S_1$ run to get $\langle LB_\mathfrak{C}\rangle^A_b \leftarrow$ SLB($\langle U\rangle^A, \langle U^2\rangle^A, \langle L\rangle^A, \langle L^2\rangle^A, \langle\mathfrak{C}^\beta\rangle^A, \langle(\mathfrak{C}^\beta)^2\rangle^A$).
  4:     If SCMP($\langle LB_\mathfrak{C}\rangle^A, \langle\Delta\rangle^A$) == 0, $S_0$ and $S_1$ run to get $\langle DTW_\mathfrak{C}\rangle^A_b \leftarrow$ SDTW($\langle X\rangle^A, \langle X^2\rangle^A, \langle\mathfrak{C}^\beta\rangle^A, \langle(\mathfrak{C}^\beta)^2\rangle^A$). Else,
     continue.
  5:     If SCMP($\langle DTW_\mathfrak{C}\rangle^A, \langle\Delta\rangle^A$) == 0, $S_b$ adds the identifier $r_\beta$ of cluster $C^\beta$ to his array $\mathbf{Arr}^C_{\alpha,b}$. Else, continue.
  6: end for
  7: $S_b$ finds each sequence $\langle\bar{Y}\rangle^A_b$ in all candidate clusters $\bar{C}^\beta$ according to the identifiers stored in his $\mathbf{Arr}^C_{\alpha,b}$.
Analysis($\langle U\rangle^A, \langle U^2\rangle^A, \langle L\rangle^A, \langle L^2\rangle^A, \langle X\rangle^A, \langle X^2\rangle^A, \langle\bar{Y}\rangle^A, \langle\bar{Y}^2\rangle^A, \langle\Delta\rangle^A$):
  1: $S_b$ initializes his array $\mathbf{Arr}_{\alpha,b}$, where $b \in \{0, 1\}$.
  2: for each sequence $\bar{Y}$ in all candidate clusters $\bar{C}^\beta$ do
  3:     $S_0$ and $S_1$ run to get $\langle LB\rangle^A_b \leftarrow$ SLB($\langle U\rangle^A, \langle U^2\rangle^A, \langle L\rangle^A, \langle L^2\rangle^A, \langle\bar{Y}\rangle^A, \langle\bar{Y}^2\rangle^A$).
  4:     If SCMP($\langle LB\rangle^A, \langle\Delta\rangle^A$) == 0, $S_0$ and $S_1$ run to get $\langle DTW\rangle^A_b \leftarrow$ SDTW($\langle X\rangle^A, \langle X^2\rangle^A, \langle\bar{Y}\rangle^A, \langle\bar{Y}^2\rangle^A$). Else, continue.
  5:     $S_0$ and $S_1$ run to get SCMP($\langle LB\rangle^A, \langle\Delta\rangle^A$) == 0. Then, $S_b$ stores the identifier of $\langle\bar{Y}\rangle^A_b$ to $\mathbf{Arr}_{\alpha,b}$, respectively.
  6: end for
  7: $S_b$ sends $\mathbf{Arr}_{\alpha,b}$ to $Q$.
  8: $Q$ obtains the unique identifiers of similar sequences, and can communicate with hospitals to obtain the sequences.
```

Fig. 6. Secure DTW-based medical time series analytic protocol.

dedicates to the computation for each hospital. For simplicity of explanation and without loss of generality, we take as an example the computation between one pair of instances.

At the very beginning, S_0 initializes a dynamic array $\mathbf{Arr}^C_\alpha$ which will store the dummy identifier r_β of the promising clusters C^β named by hospitals. Such a dummy identifier is used to prevent S_0 from knowing the real identity of C^β. Given the prepared shares of vectors and their squared values of each center $\mathfrak{C}^\beta$, U and L. S_0 and S_1 collectively execute the SLB function to obtain his share of the secure LB distance between query and each center, denoted as $\langle LB_\mathfrak{C}\rangle^A_b$, where $b \in \{0, 1\}$. Afterwards, S_0 and S_1 execute the SCMP gadget to compare $\langle LB_\mathfrak{C}\rangle^A$ with the threshold $\langle\Delta\rangle^A$. If it is larger than the threshold, the current processed cluster is directly ruled out.

For the centers whose secure LB distances are smaller than the threshold, the computation proceeds to the secure DTW calculation. S_0 and S_1 execute the SDTW function to obtain the shares of the secure DTW distance between $\mathfrak{C}^\beta$ and X, denoted as $\langle DTW_\mathfrak{C}\rangle^A_b$. They then check if $\langle DTW_\mathfrak{C}\rangle^A$ is less than $\langle\Delta\rangle^A$ via the SCMP gadget. If so, all sequences in C^β are eligible to submit to the Analysis phase for sequential scan, and its dummy identifier r_β is stored in $\mathbf{Arr}^C_\alpha$. After examining all centers, S_0 grabs all candidate sequences in the promising clusters $\bar{C}^\beta$ based on the dummy identifiers.

6.3 Secure DTW-based Analysis Phase

During the Analysis phase, S_0 and S_1 evaluate every candidate sequence $\bar{Y}$, i.e., the results of the previous phase. First, S_0 and S_1 initialize dynamic arrays $\mathbf{Arr}_{\alpha,b}$, which will store the shares of the resulting secure DTW distances, where $b \in \{0, 1\}$. Then, S_0 and S_1 execute the SLB function to obtain shares of secure LB distance between X and each $\bar{Y}$, denoted as $\langle LB\rangle^A_b$. They then test if $\langle LB\rangle^A_b$ is within the threshold via the SCMP gadget. If so, they proceed the SDTW function to obtain the shares of secure DTW distance $\langle DTW\rangle^A_b$ between X and $\bar{Y}$. Service providers then compare $\langle DTW\rangle^A$ and $\langle\Delta\rangle^A$ via the SCMP gadget to get their shares of the smaller one, and store to $\mathbf{Arr}_{\alpha,b}$. After evaluating all candidate sequences, S_0 and S_1 send the result $\mathbf{Arr}_{\alpha,b}$ to Q. Finally, the querier recovers and obtains the unique identifier of similar sequences. It can then communication with the hospitals to acquire the sequences.

TABLE 1
Number of Calls of Atomic Functions

	Pruning phase	Analysis phase
#SBranch	$2\|X\|mk$	$2\|X\|\mu mn$
#SFindMin	$\theta_{sdtw}\sigma mk$	$\theta_{sdtw}\mu\sigma mn$
#SCMP	$(1+\sigma)mk$	$(1+\sigma)\mu mn$
#SSED	$(\theta_{sdtw}\sigma+2\|X\|)mk$	$(\theta_{sdtw}\sigma+2\|X\|)\mu mn$
#SLB	mk	μmk
#SDTW	σmk	$\mu\sigma mn$

TABLE 2
Number of Beaver's Triples in Atomic Functions

(GC-based)	SSED	SLB	SDTW
$\#MTs$	1	$2\|X\|$	$\|X\|(2cr+1)-cr^2-cr$
(sharing-based)	SCMP	SBranch	SFindMin
$\#MTs$	1	3	6
# AND Triples	$3\|X\|-4$	$3\|X\|-4$	$6\|X\|-8$

6.4 Remark of Complexity

Complexity of online phases: Define the sequence length $|X|$, the parameter cr of the global constraint, the pruning ratio σ of the SLB function in the Analysis phase, and the overall pruning ratio μ of the Pruning phase. Given the number of datasets m, the total number of sequences n that each dataset contains, and the number of clusters k that each dataset is grouped into. Launching one time of the SLB function invokes $2|X|$ times of the SSED function and the SBranch gadget, respectively. While one call of the SDTW function executes $|X| \cdot (2cr+1) - cr \cdot (cr+1)$ (denoted as θ_{sdtw}) calls of the SSED function and the SFindMin gadget, respectively. The number of calls of the distance functions and the cryptographic gadgets are benchmarked in Table 1.
Workload in offline phase: The workload of the Setup phase focuses on generating Beaver's Triples (i.e., Multiplication Triples and Boolean AND Triples) as summarized in Table 2. Although the service providers can generate the Triples at offline, this workload directly reflects on the overall resource consumption. Given the total number of centers and sequences are $m \cdot (k+n)$ and $cr = 0.05|X|$. For GC-based realizations, the overall number of MTs is estimated as $M = \lceil (0.0975|X|^2 + 2.95|X|) \rceil \cdot m \cdot (n+k)$. For sharing-based realizations, the service providers can pregenerate $\lceil (0.6975|X|^2 + 6.95|X|) \cdot m \rceil \cdot (n+k)$ MTs and $\lceil (0.5975|X|^2 + 11.75|X| - 16) \cdot \rceil m \cdot (n+k)$ AND Triples. However, our protocol consumes far less than the above estimated amount. Because our pruning strategy excludes a considerable amount of the unpromising sequences (at least 40% confirmed by our experiment). As the Triples are independent on the time series data, the uncirculated one can supply for the future computations.

7 Security Analysis

We now analyze the security of the proposed protocol for secure DTW-based medical analytics. Recall that our protocol Φ involves multiple parties possessed their own datasets, where any analytic task between a single hospital and the querier can be executed in parallel by a pair of instances from the service providers S_0 and S_1. To define the security for this multi-instance and concurrent-execution protocol Φ, we follow the *Universally Composable* (UC) security framework [33]. Under a general protocol composition operation (*universal composition*), the security of Φ is preserved.

Given the engaged parties a querier Q, hospitals $H_1, ..., H_m$ and two non-colluding service providers S_0, S_1. Consider a semi-honest $\mathcal{A}$ who can corrupt only one of the two non-colluding service providers at most. We call such an $\mathcal{A}$ as an *admissible adversary*. This setting captures the property that S_0 and S_1 are non colluded. That is, if S_0 is corrupted by $\mathcal{A}$, S_1 behaves honestly; vice versa. The protocol realization based on GC and secret sharing follows directly the security of GC [11], and additive sharing [12] and Boolean sharing [13], respectively. All medical sequences, Multiplication Triples MTs, Boolean AND Triples, and intermediate results are well protected as randomly generated shares in the ring $\mathbb{Z}_{2^\ell}$ and the ring $\mathbb{Z}_2$. Given above, we argue that Φ UC-realizes an ideal functionality $\mathcal{F}$ against $\mathcal{A}$. The security captures the property that the only pertinent data learned by any corrupted party is his inputs and outputs from the protocol yet nothing about the data of the remaining honest party.

To prove the above argument, we present the details of the security definition under UC framework. Given the protocol Φ, each instance of Φ executed by the parties runs as subroutine of multiple interactive Turing Machines. Given the target ideal functionality $\mathcal{F}$. Suppose a polynomial-time semi-honest *admissible adversary* $\mathcal{A}$ who can corrupt one of the two service providers at most. The input of $\mathcal{A}$ is chose arbitrarily by a polynomial-time algorithm entity *environment machine* $\mathcal{E}$, who will also collect the outputs from the parties and $\mathcal{A}$ once the execution is terminated. In particular, $\mathcal{E}$ can exchange messages with the $\mathcal{A}$ at any time throughout the execution. Because $\mathcal{A}$ acts under the instructions of $\mathcal{E}$, we call it dummy adversary. Eventually, $\mathcal{E}$ outputs a bit. Let $\text{REAL}_{\Phi,\mathcal{A},\mathcal{E}}(\lambda, z)$ denote the output of $\mathcal{E}$ when interacting with $\mathcal{A}$ and parties running Φ on security parameter λ and uniformly chosen input z. Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{E}}(\lambda, z)$ denote the output of $\mathcal{E}$ when interacting with an ideal world adversary $\mathcal{S}$ and dummy parties running $\mathcal{F}$ on λ and z. In the ideal world, dummy parties send their inputs to $\mathcal{F}$ and forward the response to $\mathcal{E}$. We say Φ UC-realizes $\mathcal{F}$ if for any $\mathcal{A}$, there exists $\mathcal{S}$ that no $\mathcal{E}$ can determine with non-negligible probability whether it has interacted with Φ under $\mathcal{A}$ or with $\mathcal{F}$ under $\mathcal{S}$. That the following Eq 3 is negligible:

$$|Pr[\text{REAL}_{\Phi,\mathcal{A},\mathcal{E}}(\lambda, z) = 1] - Pr[\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{E}}(\lambda, z) = 1]|. \tag{3}$$

Furthermore, for a composed protocol $\Phi^{\mathcal{G}\to\rho}$, suppose a subroutine protocol ρ (gadgets and atomic operations) of Φ securely evaluates an ideal functionality $\mathcal{G}$ for $\mathcal{A}$, in the context that Φ can have multiple instances of $\mathcal{G}$ operated concurrently. The UC theorem [33] states that running $\Phi^{\mathcal{G}\to\rho}$ has the same effect of running Φ if we replace a call to an instance of $\mathcal{G}$ with a call to an instance of ρ. If Φ UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, so dose $\Phi^{\mathcal{G}\to\rho}$.

Observe that both Pruning and Analysis phases are composed with SLB, SDTW, and SCMP, except if SCMP discloses the rank between threshold and DTW. We take Analysis as an example to provide our security proof. Given the ideal functionality $\mathcal{F}_{analysis}$ defined in Fig. 7. Let $\mathcal{F}_{setup}$, $\mathcal{F}_{slb}$, and $\mathcal{F}_{sdtw}$ be the ideal functionalities for Setup, SLB, and SDTW, respectively. Let $\mathcal{F}_{scmprank}$ and $\mathcal{F}_{scmp}$ denote the ideal functionalities of the SCMP gadget with and without leaking the rank. Note that the above mentioned

Functionality $\mathcal{F}$: Interact with dummy querier Q, hospitals $H_1, ..., H_m$, service providers S_0, S_1, and an adversary $\mathcal{S}$.
Initiate: Upon receiving initiate calls from all parties, sends the message to $\mathcal{S}$, initializes a dictionary T with self-incremental index k.
Inputs: Upon receiving ($\mathsf{Input}, m+1, k, \langle X\rangle^A, \langle U\rangle^A, \langle L\rangle^A, \langle\Delta\rangle^A$) from Q, sets $T[k] = \langle X\rangle^A||\langle U\rangle^A||\langle L\rangle^A||\langle\Delta\rangle^A$; or ($\mathsf{Input}, \alpha, k, \langle Y\rangle^A$) from H_α, sets $T[k] = \langle Y\rangle^A$; or ($\mathsf{Input}, m+2, k, MTs$) from S_0, S_1, sets $T[k] = MTs$. Sends (Input, i, k) to $\mathcal{S}$.
Random: Upon receiving (random, k), randomly generate $r \in \mathbb{Z}_{2^\ell}$, sets $T[k] = r$ and sends to all parties and $\mathcal{S}$.
SLB **call:** Upon receiving (SLB, x, y, z, w), sets $T[z] = \mathcal{F}_{slb}(T[x], T[y], T[w])$ and sends (SLB, x, y, z, w) to $S_0, S_1, \mathcal{S}$.
SDTW **call:** Upon receiving ($\mathsf{SDTW}, x, y, z, w$), sets $T[z] = \mathcal{F}_{sdtw}(T[x], T[y], T[w])$ and sends ($\mathsf{SDTW}, x, y, z, w$) to $S_0, S_1, \mathcal{S}$.
SCMP **leak rank call:** Upon receiving (SCMP, x, y, z), sets $T[z] = \mathcal{F}_{scmprank}(T[x], T[y])$ and reveals (SCMP, x, y, z) to $S_0, S_1, \mathcal{S}$.
SCMP **call** Upon receiving ($\mathsf{SCMP}, x, y, z, w$), sets $T[z] = \mathcal{F}_{scmp}(T[x], T[y], T[w])$ and sends ($\mathsf{SCMP}, x, y, z, w$) to Q and $\mathcal{S}$.
Output Upon receiving (Output, z), outputs $z, T[z]$ to $\mathcal{S}$ and Q.

Fig. 7. Ideal functionality $\mathcal{F}$.

ideal functionalities can be implemented by both realizations based on GC and secret sharing, given equivalent functionalities they have achieved.

Theorem 1. *Let protocol Φ be the* Analysis *phase defined in Fig. 6. In the $\mathcal{F}_{setup}, \mathcal{F}_{slb}, \mathcal{F}_{sdtw}, \mathcal{F}_{scmprank}$, and $\mathcal{F}_{scmp}$-hybrid model, Φ UC-realizes $\mathcal{F}_{analysis}$ in Fig. 7 against a polynomial-time semi-honest admissible adversary.*

Proof. Let $\Phi^{\mathcal{G}\rightarrow\rho}$ denote the above defined composed protocol Φ. Let $\mathcal{A}$ corrupt $Q, H_1, ..., H_{m-1}$ and S_0, and interact with parties executing $\Phi^{\mathcal{G}\rightarrow\rho}$. We further define a special adversary $\mathcal{D}$ for each subroutine protocol ρ, such that there exists $\mathcal{S}_\rho$ guarantees ρ UC-realizes $\mathcal{G}$. We describe a simulator $\mathcal{S}$ for $\mathcal{A}$ in the ideal world.

$\mathcal{S}$ runs $\mathcal{A}$ and exchanges backdoor messages under the instructions of $\mathcal{A}$. It first forwards the inputs from the corrupted parties' (randomly picked by $\mathcal{E}$) to $\mathcal{F}_{analysis}$. It then randomly generates a set of additive shares in finite field $\mathbb{Z}_{2^\ell}$ on behalf of H_m, sends them to $\mathcal{F}_{analysis}$ and $\mathcal{E}$. $\mathcal{S}$ then plays the role of S_1 and interacts with $\mathcal{A}$ using randomly generated shares, excepting when it receives a call of subroutine protocol. Upon receiving a call of ρ, $\mathcal{S}$ acts as an environment for $\mathcal{S}_\rho$. Specifically, $\mathcal{S}$ forwards the backdoor messages received from $\mathcal{E}$ to $\mathcal{S}_\rho$. Upon receiving a backdoor value from $\mathcal{S}_\rho$, $\mathcal{S}$ submits the output to $\mathcal{E}$. Except the subroutine protocol SCMP gadget with leaking rank, all inputs and outputs of the protocol Φ and subroutine protocols ρ are produced directly follows the security of the additive shares and GC. To handle the revealed rank, $\mathcal{S}$ maintains a key-value stored dictionary T to record the relationship between the rank output from $\mathcal{S}_\rho$ of $\mathcal{F}_{scmprank}$ and the tuple (distance, threshold). When $\mathcal{F}_{analysis}$ receives a call of SCMP (leak rank), $\mathcal{S}$ sends to $\mathcal{S}_\rho$ its randomly selected share of distance and the threshold received from $\mathcal{A}$. To keep the consistency, $\mathcal{S}$ checks T to see if it has the key (distance, threshold). If the key exists, $\mathcal{S}$ forwards $T[(\text{distance, threshold})]$ to $\mathcal{E}$. Otherwise, upon receiving the rank produced from $\mathcal{S}_\rho$, $\mathcal{S}$ forwards the rank to $\mathcal{E}$ and saves it as $T[(\text{distance, threshold})] = \text{rank}$. As the end, we argue that the $\mathcal{E}$'s views of real and ideal world are identical. □

8 Experiments

8.1 Setup

The prototype of our secure DTW-based medical analytic protocol is implemented in Java. For our GC-based realizations, we regard the FlexSC [34] as a code base. FlexSC is a Java-based MPC toolkit that offers half-AND [35] optimizations of GC and OT extension [20]. Towards Beaver's Triples generation, we extend its OT implementation to the COT. For the additive Sharing, we set the ring as 2^{31} to fit in the Java primitive type int instead of BigInteger so as to accelerate the modular additions and multiplications.

Our experiment is conducted on the Amazon EC2 c5.4xlarge instances running Ubuntu 16.04 LTS. For demonstration purpose and without loss of generality, we launch four instances to perform the two non-collude service providers, a hospital and a querier. Each instance is equipped with a 3.0GHz Intel Xeon Platinum CPU with 16 vCPUs, 32GB of RAM, and 10Gpbs virtual NIC. It is noted that the performance of secure DTW processing is dependent on the actual size of time series datasets aggregated at the service providers, regardless of the number of hospitals.

Our experiments are based on a real medical time series dataset comprised of 15K ECG sequences which are derived from two UCR time series dataset archives, i.e., the UCR Time Series Classification Archive [36] and the UCR Suite [32]. In particular, the dataset used by us contains real-world 256Hz ECG sequences. Each sequence is comprised of 128 single-dimensional vectors (or features), which indicate the heart rates in a time period. They are normally used to evaluate the data mining problems on medical time series data [2]. In practice, they might be used to identify heart diseases, such as the Premature Ventricular Contraction (an abnormal heartbeats symptom). We store the dataset to the Redis database as key-value pairs, both in the hospital side and the service providers side during analytics. To be compatible with the secure computation primitives, we further normalize and quantize each data point to integer by scaling up 105 times. For the lower bounding distance and DTW, we employ the global constraint as a system parameter. Following the setting in [2], we set its value to $cr = 0.05 * 128 = 7$.

To improve runtime performance, we catch the intermediate results in Redis database. We observe that the heap size of the Java Virtual Machine (JVM) is insufficient to store our storage-consumed intermediate results, such as the all-pair distance matrix used for DP clustering. In detail, consider the 64-bit JVM. By default all variables and intermediate results are stored in the heap of JVM at runtime. Although the heap size is 2^{64} bits in theoretical, it actually depends on how operating system allocates, because no system so far has 2^{64}-bit RAM. In fact, we observe that the maximum JVM

TABLE 3
Performance of Cryptographic Gadgets

	GC based realization			Sharing based realization			
Gadgets	Time (ms)	Comm. (KB)	# AND gates	Time (ms)	Comm. (KB)	# AND Triples	# MTs
SCMP	4.65	22.29	541	1.95	0.38	89	1
SCMP (leak rank)	3.95	17.63	350	1.85	0.35	89	0
SBranch	6.11	31.62	859	2.18	0.44	89	3
SFindMin	5.53	28.2	764	4.60	0.88	178	6

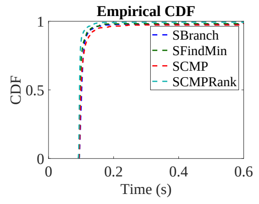

Fig. 8. Unit time for GC based gadgets.

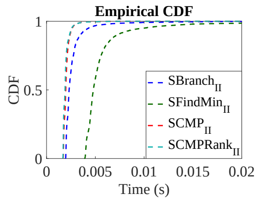

Fig. 9. Unit time for sharing based gadgets.

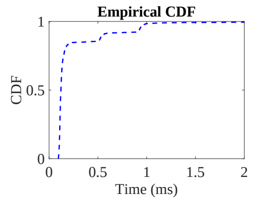

Fig. 10. The SSED function unit time.

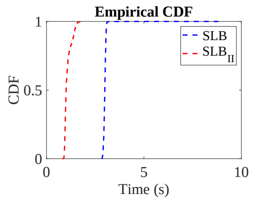

Fig. 11. The SLB function unit time.

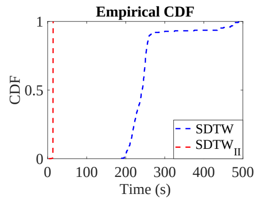

Fig. 12. The SDTW function unit time.

heap size in our Ubuntu OS is about 7.6GB, much smaller than the 32GB RAM. As a result, we treat the in-memory Redis database (v3.0.6) as software glue linking up sub-modules of our prototype to catch the intermediate results.

In addition, we generate a bunch of Multiplication Triples and Boolean AND Triples offline. We store them in files that 300 triples for each. Once required, the service providers can randomly choose files and retrieve the triples upon the demand of computation, and then delete the files to ensure the randomization of triples Besides, we slightly extend FlexSC toolkit to allow a single call of the SLB function to be executed by multiple threads.

It is noted that the assessment of data quality in machine learning is a research topic orthogonal to this paper. Our focus is on providing a privacy-preserving solution to a data mining problem, i.e., DTW-query computation on medical time series data. Our secure system builds on intact plaintext DTW processing techniques including the DTW distance, the Keogh's Lowering Bounding distance [18], and the Density Peak clustering [3], [17]. The correctness of our solution is ensured by the underlying cryptographic primitives.

8.2 Evaluation

8.2.1 Cryptographic Gadgets

Table 3 benchmarks the unit latencies and bandwidths of the gadgets on average by 10^4 executions in LAN. For the GC-based realizations, the reported times exclude the offline circuits generation times, and we also report the number of AND gates for unit execution. While for the sharing-based realizations, we report the numbers of Beaver's Triples consumed in each call. The prominent takeaway is that, the unit bandwidth costs of the sharing-based realizations are 30-70$\times$ reduced than the equivalent GC-based ones. We take the bandwidth cost of executing 10^6 times the SBranch gadget for 5K sequences as an example. The sharing-based realization saves the bandwidth from 30 GB to 0.3 GB compared with the GC-based one. Such a saving reduces the overall resources demand of medical service providers and results in financial benefits.

On the other hand, the GC-based realizations are more suitable for the high-latency WAN network due to its constant-round property. For demonstration purpose, we simulate a WAN network with an average 50ms delay per communication.The result shows that the GC-based realizations are faster than the sharing-based realizations. For example the SCMP gadget, the unit latency of the GC-based realization is 0.85s and sharing-based realization is 9.55s.

Moreover, we leverage the empirical cumulative distribution function (CDF) to depict the distributions of the unit running times for each gadget in LAN. Fig. 8 illustrates the distribution of unit times on executing 1K gadgets realized based on GC. Statistically, most of the four gadgets can be proceeded within 0.2 seconds. Fig. 9 displays the empirical CDF of unit times for the sharing-based realizations. It shows that the SCMP gadget (with and without revealing the rank), the SBranch gadget, and the SFindMin gadget can be executed within 2.5ms, 5ms, and 15ms, respectively.

8.2.2 Distance Functions

Table 4 benchmarks the performances of executing individual distance functions SSED, SLB, and SDTW, with an average of 10^4 executions in LAN. For the SSED function, both time and bandwidth are lightweight. For the SLB function, it invokes the SBranch gadget to perform vector-wise secure branching. The sharing-based SLB achieves around 100$\times$ saving on the bandwidth consumption. Likewise, the SDTW function invokes the SFindMin gadget. The sharing-based SDTW introduces by about 20$\times$ speed up in LAN and 45$\times$ bandwidth saving than the GC-based SDTW.

Besides, we depict the empirical CDF to illustrate the unit runtime distributions over 5K executions of the above distance functions. Fig. 10 shows that, in most cases, the unit time for the SSED function is less than 1.5ms. Fig. 11 exhibits that, 4s is sufficient to proceed a single call of GC-based SLB function, and 2s is enough to perform the sharing-based realization (i.e., SLB$_{II}$). Fig. 12 illustrates the distribution of unit time of the SDTW function. For the GC-based one, it requires at least 200s, while retrenching to less than 20s when employing the sharing-based one (i.e., SDTW$_{II}$).

8.2.3 Online Phases

We turn our focus on the performance in each phase of our protocol. The overall running times of online phases (i.e., the Pruning phase + the Analysis phase) are evaluated on 5K, 10K, and 15K sequences. Each set of sequences is randomly

TABLE 4
Performance of Atomic Distance Functions

	GC based realization		Sharing based realization	
Functions	Time	Comm.	Time	Comm.
SSED	-	-	1.88 ms	0.03 KB
SLB	0.84 s	11.13 MB	1.61 s	0.11 MB
SDTW	216.76 s	74.95 MB	9.97 s	1.63 MB

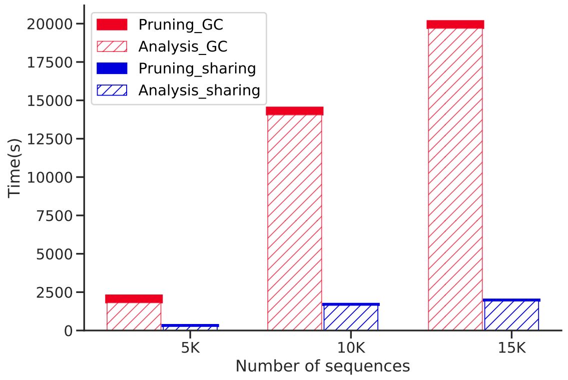

Fig. 13. Overall runtime for online phases.

chosen from our dataset and already grouped in 10 clusters. For demonstration purpose, we choose at random one of the secure DTW distance 22834481 as our threshold. As a result, there are 8 sequences out of 5K data, 59 out of 10K data, and 87 out of 15K data similar to the query.

Fig. 13 depicts, along with the growth of dataset in chunk of 5K sequences, the time of Pruning phase remains steady for both realizations, since its workload relies upon the number of clusters. Whereas for the time of Analysis phase, the time growths sharply yet does not form a linear scale. Because its overhead depends on two perspectives: (1) the sequential scan with all candidate sequences submitted to Analysis phase via the SLB function, and (2) the comparison based on the quadratic time SDTW function with the above results. Given the result, the sharing-based realization achieves up to 90% saving in LAN compared with the GC-based realization. In particular, the time for analyzing 15K sequences is reduced from 5.6 hours to 34 minutes.

To verify the effectiveness of the Pruning phase, we define the pruning ratio as the number of excluded sequences divides the size of dataset (i.e., 15K sequences). We randomly choose a group of thresholds as t1: 10511173, t2: 18848334, and t3: 22834481 for demonstration purpose. We report that the ratios 96.3%, 93.2%, 91.0% for t1, 71.6% 69.5% 68.2% for t2, and 50.0% 43.9% 41.5% for t3 when the dataset are partitioned into 10, 20 and 30 clusters. The results show that the pruning ratios decrease with the raising number of clusters and values of threshold, yet diminishing at least 6K sequences from the sequential scan in the Analysis phase.

8.2.4 Setup Phase

The workload of the querier is lightweight. For 1K queries, it takes 51.5ms to generate U, L, and 38.5ms to generate shares. The result confirms that our protocol is amiable to the querier who does not have strong computational power. The workload of the hospital in the offline phases is dominated by the one-time DP clustering [17]. It takes 5467.6s, 55558.2s, and 421895.4s to classify 5K, 10K, and 15K sequences into 10 clusters. The runtime scales quadratically with the growth amount of sequences, due to the intensive workload of calculating all-pair distances. But the Setup phase does not aggravate the workload of the hospital: the shares of 15K sequences can be produced within 1s. The workload of two service providers in Setup phase is triple generation. Generating one triple consumes 6.1ms and 30.4KB on average by 10^5 executions. To save the time and bandwidth costs, one may resort to an independent third party dedicated to preparing the triples [16], [37].

9 Related Works

DTW-based Time Series Data Analytics: The DTW technique has been pervasively used in data mining domains related with time series data, including biomedicine [3], image/speech processing [38], and astronomy. The DTW distance is firstly proposed by Berndt *et al.* [39]. Since then, the research efforts in the literature have been devoted to improving the efficiency and effectiveness. To accelerate the DTW-query computation, Keogh *et al.* [18] propose an indexing method based on the lower bounding distance to prune off unpromising time series sequence prior to the DTW distance calculation. Atop this index design, a suite of optimization techniques [2] has been proposed to support search over trillions of streaming time series data. In the meantime, approximated DTWs [38], [40], [41] are also proposed to speed up the search. Constraint methods on DTW can be adopted to avoid pathological mapping of two time series, such as Sakoe-Chiba band [31]. To support multi-dimensional time series sequences, the work [42] generalizes the conventional DTW to enhance the accuracy. Beyond search, DTW can also be applied to time series classification [43] and clustering [3]. Despite the benefits, these works do not consider privacy protection.

Privacy-Preserving Analytics over Medical Data: Another related line of research is privacy-preserving medical data analytics for different applications. In [23], Zheng *et al.* design a privacy-preserving medical image denoising system based on Deep Neural Network (NN). Their system produces high-quality content for image-centric applications (e.g., analyzing Chest X-ray images), while enabling a privacy assured remote diagnosing. In [7], Barni *et al.* design a privacy-preserving ECG data classification system via linear branching program and NN. Some works [44], [45], [46] design secure similarity search schemes over outsourced and encrypted medical databases. There are also some studies on privacy-preserving human genomic data analytics. The work [47] securely computes Euclidean distance and Pearson correlation coefficient distance on human genome sequences, yet the adopted distance metrics are error-prone when any deletions and additions happen on the genome sequences. The privacy-preserving DNA sequence evaluation systems proposed in [48] and [49] are based on the robust Edit Distance (ED). Similar to DTW, the ED is relatively complicated and evaluated via dynamic programming. These systems rely on a public-available human genomic reference dataset in cleartext, so as to approximate and transform the secure ED computation to a simpler problem. Other secure applications of genomics are also addressed like secure genome-wide association studies [50]. In [51], Zhu *et al.* propose a homomorphic encryption based protocol that allows a client and a server, each holding a time series sequence in the clear, to compute and reveal the DTW distance to both them. Their scenario is quite different from the far more challenging one we target in this paper, i.e., privacy-preserving collaborative analytics over distributed and encrypted time series data. To our best knowledge, no prior work has addressed the challenges as we did in this paper.

Privacy-Preserving Machine Learning: The area of privacy-preserving machine learning has rapidly emerged in recent years. Some systems have been proposed based on the generic secure computation techniques for different machine learning problems, such as privacy-preserving ridge-regression [52], privacy-preserving association rule mining [53], and privacy-preserving NN training and inference system [25]. Due to the prohibitively high performance overhead introduced by the generic MPC techniques, the current trend is to leverage problem specifics and develop tailored protocols. For instance, the works [14], [54], [55] design cryptographic protocols customized for privacy-preserving neural network inference. These works aim to enable a client holding a private input and a server holding a proprietary neural network model to run a secure protocol, so that the client only learns the inference result while the server learns nothing. Also, the client-server setting considered in these works is relatively simpler than a decentralized analytic setting. In other independent work, Zheng *et al.* [16] propose a privacy-preserving decision tree inference system with tailored and lightweight cryptography techniques in the two-server model. The systems Quotient [15] and Leia [56] are designed to support oblivious NN inference via a co-design of machine learning and cryptography techniques in the two-server model. Inspired by the trend in privacy-preserving machine learning, we explore and present new endeavors for a secure data mining problem. That is, we design the first system tailored for privacy-preserving DTW-based collaborative analytics on distributed medical time series data, where we make use of observations from both data mining and cryptography domains to accomplish a scalable and secure design.

Difference from Conference Version: Portions of the work presented in this paper have been presented in [1]. We have revised the preliminary work [1] a lot and made substantial new contributions and improvements, which are summarized below. Firstly, we have proposed new efficient secret sharing-based designs in Section 5 for the secure distance functions and cryptographic gadgets essential for privacy-preserving DTW-based medical time series data analytics. Such new designs lead to the provision of a new alternative protocol that endows our system with the flexibility and capability to cater for different realistic service demands in practice. Secondly, we have justified the security guarantees of our designs in Section 7 with formal security proofs. In particular, we have defined the ideal functionality for our target service of privacy-preserving DTW-based medical data analytics. Based on the UC security framework, we have formally proved that our protocol UC-realizes the ideal functionality in a hybrid model. Thirdly, we have significantly improved the experiments, especially with new extensive evaluation on the new secure secret sharing-based designs. The results show that in the LAN environment the new realization of our security design substantially improves upon the one in [1] using garbled circuits by up to $10\times$ in online runtime. Finally, we have substantially polished the presentation of the whole paper to reflect our new contributions and latest understanding on the topic, as well as improve the clarity.

10 Conclusion

In this paper, we present the first system design of privacy-preserving DTW-based analytics on distributed medical time series data, which allows geographically separated healthcare institutions to collaborate in a DTW-based medical analytics service like disease screening for public health. Our design is built from a synergy of techniques from both cryptography and data mining domains, where we uniquely bridge effective hybrid pruning techniques with efficient secure computation techniques to develop a tailored solution. Formal security analysis is provided to justify the security guarantees of our design. Our comprehensive evaluation demonstrates the practically affordable performance of our system for secure DTW-based medical analytics.

Acknowledgement

This work was supported in part by the ARC Linkage Project LP160101766: Privacy-preserving cloud data mining-as-a-service.

References

[1] X. Liu and X. Yi, "Privacy-preserving collaborative medical time series analysis based on dynamic time warping," in *Proc. of ESORICS*. Springer, 2019.

[2] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. of ACM SIGKDD*, 2012.

[3] N. Begum, L. Ulanova, J. Wang, and E. Keogh, "Accelerating dynamic time warping clustering with a novel admissible pruning strategy," in *Proc. of ACM SIGKDD*, 2015.

[4] W. Zheng, R. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *Proc. of IEEE S&P*, 2019.

[5] C. Wang, B. Zhang, K. Ren, J. M. Roveda, C. W. Chen, and Z. Xu, "A privacy-aware cloud-assisted healthcare monitoring system via compressive sensing," in *Proc. of IEEE INFOCOM*, 2014.

[6] Y. Lindell and B. Pinkas, "Privacy preserving data mining." *Journal of cryptology*, vol. 15, no. 3, 2002.

[7] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Trans. on Information Forensics and Security*, 2011.

[8] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPPA)," online at https://www.hhs.gov/hipaa/index.html, 1996.

[9] European Parliament and of the Council, "The General Data Protection Regulation (GDPR)," online at http://data.europa.eu/eli/reg/2016/679/2016-05-04, 2016.

[10] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *Proc. of NDSS*, 2015.

[11] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. of IEEE FOCS*, 1986.

[12] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proc. of WPES*, 2004.

[13] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," in *Proc. of ACM STOC*, 1987.

[14] "Delphi: A cryptographic inference service for neural networks," in *Proc. of 29th USENIX Security*, 2020.

[15] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón, "Quotient: two-party secure neural network training and prediction," in *Proc. of ACM CCS*, 2019.

[16] Y. Zheng, H. Duan, and C. Wang, "Towards secure and efficient outsourcing of machine learning classification," in *Proc. of ESORICS*. Springer, 2019.

[17] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[18] E. Keogh, "Exact indexing of dynamic time warping," in *Proc. of VLDB*, 2002.

[19] "Physiobank atm," Online at http://physionet.org/cgi-bin/atm/ATM.

[20] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proc. of ACM CCS*, 2013.

[21] Y. Lindell and B. Pinkas, "A proof of security of yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.

[22] M. O. Rabin, "How to exchange secrets with oblivious transfer," *Harvard University, Tech. Rep. TR-81*, 1981.

[23] Y. Zheng, H. Duan, X. Tang, C. Wang, and J. Zhou, "Denoising in the dark: Privacy-preserving deep neural network based image denoising," *IEEE Trans. on Dependable and Secure Computing*, 2019.

[24] W. Chen and R. A. Popa, "Metal: A metadata-hiding file sharing system," in *Proc. of NDSS*, 2020.

[25] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Proc. of IEEE S&P*, 2017.

[26] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Trans. Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[27] Y. Zheng, H. Duan, and C. Wang, "Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing," *IEEE Trans. on Information Forensics and Security*, vol. 13, no. 10, pp. 2475–2489, 2018.

[28] J. Brickell and V. Shmatikov, "Efficient anonymity-preserving data collection," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 76–85.

[29] P. Syverson, R. Dingledine, and N. Mathewson, "Tor: The second-generation onion router," in *Usenix Security*, 2004, pp. 303–320.

[30] F. Tramèr, F. Zhang, H. Lin, J. Hubaux, A. Juels, and E. Shi, "Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge," in *Proc. of IEEE EuroS&P*, 2017.

[31] H. Sakoe, S. Chiba, A. Waibel, and K. Lee, "Dynamic programming algorithm optimization for spoken word recognition," *Readings in speech recognition*, vol. 159, p. 224, 1990.

[32] "The ucr suite," Online at http://www.cs.ucr.edu/~eamonn/UCRsuite.html.

[33] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," Cryptology ePrint Archive, Report 2000/067, 2000.

[34] X. Wang, "Flexsc," Online at https://github.com/wangxiao1254/FlexSC, 2018.

[35] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in *Proc. of Eurocrypt*, 2015.

[36] "Ucr time series classification archive," Online at https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[37] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. of AsiaCCS*, 2018.

[38] Y. Zhang and J. R. Glass, "An inner-product lower-bound estimate for dynamic time warping," in *Proc. of IEEE ICASSP*, 2011.

[39] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. of KDD workshop*, 1994.

[40] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "Ftw: fast similarity search under the time warping distance," in *Proc. of ACM PODS*, 2005.

[41] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[42] M. Shokoohi-Yekta, J. Wang, and E. Keogh, "On the non-trivial generalization of dynamic time warping to the multi-dimensional case," in *Proc. of SDM*, 2015.

[43] Y. Chen, B. Hu, E. Keogh, and G. E. Batista, "Dtw-d: time series semi-supervised learning from a single example," in *Proc. of ACM SIGKDD*, 2013.

[44] L. Xu, S. Sun, X. Yuan, J. K. Liu, C. Zuo, and C. Xu, "Enabling authorized encrypted search for multi-authority medical databases," *IEEE Trans. on Emerging Topics in Computing*, 2019.

[45] X. Yuan, X. Wang, C. Wang, J. Weng, and K. Ren, "Enabling secure and fast indexing for privacy-assured healthcare monitoring via compressive sensing," *IEEE Trans. on Multimedia*, vol. 18, no. 10, pp. 2002–2014, 2016.

[46] X. Liu, X. Yuan, and C. Wang, "Encsim: An encrypted similarity search service for distributed high-dimensional datasets," in *Proc. of IEEE/ACM IWQoS*, 2017.

[47] A. Salem, P. Berrang, M. Humbert, and M. Backes, "Privacy-preserving similar patient queries for combined biomedical data," *Proc. of PETS*, 2019.

[48] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proc. of ACM CCS*, 2015.

[49] T. Schneider and O. Tkachenko, "Episode: Efficient privacy-preserving similar sequence queries on outsourced genomic databases," in *Proc. of ACM AsiaCCS*, 2019.

[50] O. Tkachenko, C. Weinert, T. Schneider, and K. Hamacher, "Large-scale privacy-preserving statistical computations for distributed genome-wide association studies," in *Proc. of ACM AsiaCCS*, 2018.

[51] H. Zhu, X. Meng, and G. Kollios, "Privacy preserving similarity evaluation of time series data." in *Proc. of EDBT*, 2014.

[52] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. of IEEE S&P*, 2013.

[53] L. Li, R. Lu, K.-K. R. Choo, A. Datta, and J. Shao, "Privacy-preserving-outsourced association rule mining on vertically partitioned databases," *IEEE Trans. on Information Forensics and Security*, vol. 11, no. 8, pp. 1847–1861, 2016.

[54] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "Gazelle: A low latency framework for secure neural network inference," in *Proc. of 27th USENIX Security*, 2018.

[55] M. S. Riazi, M. Samragh, H. Chen, K. Laine, K. Lauter, and F. Koushanfar, "Xonn: Xnor-based oblivious deep neural network inference," in *Proc. of 28th USENIX Security*, 2019.

[56] X. Liu, B. Wu, X. Yuan, and X. Yi, "Leia: A lightweight cryptographic neural network inference system at the edge," Cryptology ePrint Archive, Report 2020/463, 2020, https://eprint.iacr.org/2020/463.

Xiaoning Liu is currently a Ph.D. student at the School of Computer Science and Software Engineering, RMIT University. She received her B.E. degree in computer science and technology from Henan University in 2012, and the M.S. degree in computer science from City University of Hong Kong in 2013. Her research interests include privacy-preserving data mining and machine learning.

Yifeng Zheng is currently a postdoc at City University of Hong Kong, Hong Kong. He received the PhD degree in computer science from City University of Hong Kong, Hong Kong, in 2019, and the B.E. degree in information engineering from South China University of Technology, Guangzhou, China, in 2013. He worked as a postdoc from 2019 to 2020 at Commonwealth Scientific and Industrial Organization (CSIRO), Sydney, Australia. His current research interests are focused on security and privacy related to cloud computing, IoT, machine learning, and multimedia.

Xun Yi is currently a professor with the School of Science, RMIT University, Australia. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and privacy-preserving data mining. He has published more than 150 research papers in international journals, such as the IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Wireless Communications, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Circuits and Systems, IEEE Transactions on Vehicular Technology, IEEE Communication Letters, IEE Electronic Letters, and conference proceedings. He has ever undertaken program committee members for more than 30 international conferences. Recently, he has led a few of Australia Research Council (ARC) Discovery Projects. Since 2014, he has been an associate editor of the IEEE Transactions on Dependable and Secure Computing.

Surya Nepal is a Senior Principal Research Scientist at CSIRO's Data61. He currently leads the distributed systems security group. His main research focus is on the development and implementation of technologies in the area of distributed systems (including cloud, IoT and edge computing) and social networks, with a specific focus on security, privacy, and trust. He has more than 200 peer-reviewed publications to his credit. He has co-edited three books, including security, privacy, and trust in cloud systems by Springer, and co-invented three patents. He is a member of the editorial boards of IEEE Transactions on Service Computing, ACM Transactions on Internet Technology and Frontiers of Big Data- Security Privacy, and Trust. He is currently a theme leader of the cyber security Cooperative Research Centre (CRC), a national initiative in Australia. He holds conjoint faculty position at UNSW and an honorary professor position at Macquarie University.