Privacy-Preserving Collaborative Medical Time Series Analysis based on Dynamic Time Warping

Xiaoning Liu and Xun Yi

RMIT University, Australia {maggie.liu, xun.yi}@rmit.edu.au

Abstract. Evaluating medical time series (e.g., physiological sequences) under dynamic time warping (DTW) derives insights assisting biomedical research and clinical decision making. Due to the natural distribution of medical data, a collaboration among multiple healthcare institutes is required to carry out a reliable and quality medical judgment. Yet sharing medical data cross the boundaries of multiple institutions faces widespread privacy threats, along with increasingly stringent laws and privacy regulations nowadays. Addressing such demands, we propose a privacy-preserving system tailored for the DTW-based analysis over the decentralized medical time series sequences. Our system constructs a secure and scalable architecture to deliver comprehensive results from a joint data analytic task with privacy preservation. To accelerate complicated DTW query processing, our system adapts the advancement in secure multi-party computation (MPC) framework to realize encrypted DTW computation, decomposing complicated and iterative operations into atomic functions under suitable MPC primitives and optimized for DTW. Moreover, our system introduces a secure hybrid pruning strategy that diminishes the volume of time series sequences that are submitted before and processed within the encrypted DTW query. We implement a prototype and evaluate its performance on Amazon Cloud. The empirical evaluation demonstrates the feasibility of our system in practice.

1 Introduction

Medical time series analysis produces comprehensive knowledge for modern medical research, driven by the ubiquity of medical data continuously captured by electrical sensors over time. A typical use case comes for a joint disease screening for public health, where researchers, hospitals, and healthcare institutes wish to collaboratively find patients who display similar medical characteristics to the sample of interest. To derive a reliable and quality conclusion, such an analytic task often requires specific matching algorithms over time series. As a well-known distance metric, dynamic time warping (DTW) is ascendant in answering medical time series mining, such as detecting Premature Ventricular Contraction (PVC) with Electrocardiography (ECG) data [29], and Cardiac Tamponade with Photoplethysmogram (PPG) data [10]. It is effective to handle time shifting that two time series with similar shapes will be matched even they are not synchronized in the time axis.

Advancement in the DTW-based medical analysis system makes it plausible. However, a consensus in practice has emerged that the adoption of the above system will be heavily stumbled due to the privacy issues. Unauthorized exposure of the confidential patient records inflicts severe commercial damages and putting the individuals' privacy in danger [8, 12, 25]. Atop protecting data confidentiality, underpinning medical analysis to be practical requires the collaboration of multiple institutions. For most medical practices, the data volume and diversity accumulated in a single hospital cannot provide sufficient disease information due to the intrinsic distribution of medical data [12,37]. This is challenging since privacy regulations and laws (e.g., HIPAA in USA [3] and GDPR in Europe [21]) prevent sensitive medical data from ever being shared or pooled together.

One promising approach to deal with joint medical research is leveraging the secure multi-party computation (MPC) techniques [12, 19, 33]. While the latest studies [20, 27] display the ability of MPC to handle large-scale data, how to adapt it to private time series evaluation is unclear. Meanwhile, applying MPC techniques to DTW faces with cumbersome computational overhead. DTW, as a dynamic programming algorithm, the rationale behind is to compute the all-pair underlying distance between elemental vectors of time series. Afterwards, it iteratively finds the minimum cumulative distance of a slightly larger portion of time series until reaching the entire time series. Plenty of iterations of vectorized operations introduce heavy computational costs, which may result in long processing time. As seen, the plaintext algorithm of DTW between only two time series already involves a bunch of complicated operations that are quadratic to the sequence length. Therefore, how to efficiently compute DTW in the encrypted domain, and how to build a secure architecture to facilitate healthcare institutes to jointly and scalably perform encrypted DTW over decentralized medical time series become particularly challenging.

Contributions: In this paper, we propose a system tailored for privacy-preserving collaborative medical time series data analysis based on dynamic time warping. Our system suffices for capturing the above demands: embracing the large volume medical time series sequences that are naturally decentralized, conducting secure DTW queries with practical performance, and delivering quality analytic results benefited from joint mining. The contributions of our proposed system are summarized as follows:

- We construct an architecture allowing multiple healthcare institutes to carry out a joint analytic task over encrypted medical time series sequences supplied by geographically separated parties. This architecture is amiable for a real-world medical analysis scenario. It provides a scalable and dedicated computation service to serve for multiple participants, and releases each party from heavy computation and communication workload.
- We devise a mixed protocol which modularly composes a bunch of customized atomic functions under MPC primitives. Each function in the protocol is carefully devised for the DTW algorithm. This synergy enriches the expressive power of our medical system while providing guaranteed security for the sensitive data sequences.

- Our system elaborates a hybrid pruning strategy to accelerate the secure DTW-query processing on two aspects. Globally, it employs a two-phase scheme to diminish the volume of sequences that will be submitted to the secure protocol. In the first phase, the query is compared only with cluster centers which are precomputed on local, so as to find out candidate clusters whose records are similar to the query. Afterward, in the second phase, the secure DTW query is conducted only within the resulting clusters of the first phase. From the aspect of distance algorithm, resorting to a highly parallel lower bounding technique, our system prunes off the sequences that are not possible to be the best match before submitting to heavyweight secure DTW computation.
- We implement a Java prototype of our system from FlexSC [34], and provide empirical evidence to confirm the practicality of our system using realistic public physiological sequences (ECG data) [1]. We conduct a comprehensive set of evaluations on each component and each phase of our protocol in terms of time and communication costs. Theoretically, each call of comparing two data sequences under the DTW distance function requires 1920 calculations of underlying distance and comparisons to find the minimum distance. Our results show that it takes about 2×10^4 s to process a secure DTW query over 15,000 sequences (each contains 128 vectors), achieving a 100× saving compared with naive sequential scan.

The rest of the paper is organized as follows. Section 2 discusses related literature. Section 3 introduces the preliminaries used in this paper. Section 4 presents our architecture, and detailed system and protocol designs. We give implementation and evaluation in Section 5, and conclude the paper in Section 6.

2 Related Works

We organize this section as prior arts of privacy-preserving medical data analysis, MPC frameworks and applications, and works of DTW in database domain.

Privacy-Preserving Analysis over Medical Data: Privacy-preserving analysis over human genome sequences is a long-studied problem in secure medical data mining domain, such as evaluating similarity via private set intersection [7] and distance calculation [31], and obtaining statistics via secure genome-wide association studies [19,32]. Recent design [33] proposed by Wang *et al.* embraces million-scale genomic data evaluated under private Edit Distance (ED). Their design preprocesses the data into sets and compares the plaintext values with a public reference genome to find the variances. Then by leveraging the certain pattern of human genome sequences, it approximates the ED computation as *set different size* protocols. Their secure computation is performed over the variances sets, thus achieving high scalability. Zheng *et al.* enable a private medical image (Chest X-ray images) denoising framework based on Deep Neural Network [38]. Their framework securely delivers high-quality content assuring the reliability of image-centric applications, such as cloud side diagnosing. Privacy-aware evaluation over physiological data brings new insights to the secure medical data mining, such as ECG data classification via branching program and

neural network [8]. Recently, Zhu *et al.* propose two privacy-preserving protocols to evaluate Paillier-encrypted time series data under DTW and Discrete Frechet Distance (DFD), respectively, in the client-server setting [40]. Their protocols introduce considerable amount of crypto-operations and several rounds of data transfer. Thus, the design is not suitable for the real-world collaborative mining scenario, where multiple parties are involved and the communication cost dominates the overall efficiency. This limitation enlightens us to devise a more scalable and efficient time series evaluation system for encrypted medical data.

Secure Multi-Party Computation Framework: Recent years have witnessed a paradigm shift in MPC frameworks, introducing an approach that is mixed with various MPC primitives to achieve efficient secure computation. Kerschbaum *et al.* [24] design a framework combining Yao's Garbled Circuits (GC) and homomorphic encryption (HE). Sharemind [14] develops a high-level language SecreC for Arithmetic sharing, and later is extended to a mixed protocol in [13]. A very recent framework ABY [20] provides automated generation of mixed protocols supporting efficient conversions between Arithmetic sharing, Boolean sharing and Yao's GC. Due to scalability, the mixed-protocols have been applied to many applications [15, 22, 27, 28, 39]. Following the same philosophy, we tailor a mixed protocol, one of the key components in our system, to process secure DTW queries for distributed medical data. In the meantime, Blanton *et al.* devise a general distributed platform enabling private medical data analysis [12]. They take the parents assignment problem as an example to show how their platform provides guaranteed security under different settings of the problem and the roles played by the participants. Besides, threshold HE can be used for MPC applications, such as coopetitive learning against a malicious adversary [37] and user profile matching in social networks [36].

Dynamic Time Warping for Time Series Data: DTW is a prevalent distance measure in time series data related mining domains, such as medicine, image/speech processing, and astronomy. It is firstly introduced by Berndt *et al.* in [11]. To accelerate DTW, Keogh proposes an indexing method [23] to perform similarity search on archived data. Atop this index, they design a suite of optimization techniques [29] to search on trillions of streaming data. Beyond search, DTW can also be applied to time series classification [18] and clustering [10].

3 Background

Dynamic Time Warping: Dynamic time warping [23] is a distance metric which measures the dissimilarity over time series data. It is effective to handle time shifting, whereby two time series with similar wavelets are matched even if they are "shrank" or "stretched" in the time axis.

Let $X = (\mathbf{x}_1, ..., \mathbf{x}_{|X|})$ and $Y = (\mathbf{y}_1, ..., \mathbf{y}_{|Y|})$ be two time series sequences consisting with $|X|$ and $|Y|$ numbers of dim-dimensional vectors $\mathbf{x}_i$ and $\mathbf{y}_j$, respectively. Let X_i be the subsequence of X starting from the first vector $\mathbf{x}_1$ to the i-th vector $\mathbf{x}_i$; likewise, Y_j is the subsequence of Y from $\mathbf{y}_1$ to $\mathbf{y}_j$. In our paper, we consider only the data points in X and Y being integers. Without loss of generality, we compare equal-length sequences, that is $|X| = |Y|$. We further denote a dataset contains n-sequences as $\mathbf{Y} = \{Y^1, ..., Y^n\}$.

To align X and Y, we define an *optimal warping path* indicating the minimum warping cost. It is a monotonic and non-overlapped warping path $W = w_1, \ldots, w_K$, and has to use every index of each time series. W can be evaluated via a dynamic programming approach, i.e., recursively finding the minimum cumulative distance $D(X_i, Y_j)$ of slightly larger portions of subsequences until reaching the entire sequences. $D(X_i, Y_j)$ is declared as the underlying distance $dist(\mathbf{x}_i, \mathbf{y}_j)$ (i.e., the Euclidean distance (ED)) plus the minimum of cumulative distances of adjacent cells. We formulate it as follows:

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

The $DTW(X, Y)$ is equal to $D(X_{|X|}, Y_{|Y|})$, and the optimal W is found in the reverse order by a greedy search.

DTW normally applies a global constraint to avoid pathological warping, where a relatively small portion of one sequence would not be warped to map a considerably large portion of another. The constraint introduces a cr-width sliding window that only the elements within the window can be compared; that is for $w_k = (i, j)_k$, i, j should be $j - cr \leq i \leq j + cr$. Both the time and space complexity of DTW is $O(|X||Y|)$. In our design, we expect the proposed protocol to output DTW yet hiding the optimal path since it is the intermediate result. Hence, only the cumulative distances of the adjacent cells are required to be maintained in each iteration. As a result, a linear space complexity can be achieved by only maintaining $\{D(X_i, Y_j)\}_{j=1}^{|Y|}$ and $\{D(X_{i-1}, Y_j)\}_{j=1}^{|Y|}$ in memory.

Keogh's Lower Bound of DTW: Since calculating DTW is very time-demanding, we use the linear-time lower bounding LB_{Keogh} algorithm [23] to prune off sequences that are not possible to be the best match. Given a query X, LB_{Keogh} defines an upperbound U and a lowerbound L surrounding it as $\mathbf{u}_i = max(\mathbf{x}_{i-cr} : \mathbf{x}_{i+cr})$ and $\mathbf{l}_i = min(\mathbf{x}_{i-cr} : \mathbf{x}_{i+cr})$ that $\forall_i, \mathbf{u}_i \geq \mathbf{x}_i \geq \mathbf{l}_i$. Given candidate Y, the distance $LB_{Keogh}(X, Y)$ is formulated as follows:

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

It is provably tight that $LB_{Keogh}(X, Y) \leq DTW(X, Y)$ and holds linear time complexity. LB_{Keogh} can be deemed as the ED between Y and the closer one of $\{U, L\}$. Since the function of ED is monotonic and concave, we *omit the step of square root* for the ease of deploying to secure computation.

Density Peaks Clustering: Density Peaks (DP) algorithm [30] is a density-based clustering algorithm that is amiable for time series with various shapes (unlike R-tree based DBSCAN and k-means). Given a dataset $\mathbf{Y}$ and a matrix containing all-pair DTW distances, the DP elects k cluster centers $\{\mathfrak{C}^1, ..., \mathfrak{C}^k\}$, where each $\mathfrak{C}$ is surrounded by lower local density neighbors and is relatively far from any points with higher local densities.

Arithmetic Sharing: On input an ℓ-bit value x, Arithmetic Sharing [6, 20] generates shares $\langle x \rangle_0^A, \langle x \rangle_1^A$ in the ring $\mathbb{Z}_{2^\ell}$ uniformly at random, where $\langle x \rangle_0^A +$

$\langle x \rangle_1^A \equiv x \pmod{2^\ell}$. Unless an explicit claim, *all operations performed under* $\pmod{2^\ell}$. For our shared *dim*-dimensional vectors, P_i calculates the addition non-interactively as $\langle \mathbf{z} \rangle_i^A = \langle \mathbf{x} \rangle_i^A + \langle \mathbf{y} \rangle_i^A$. Multiplication relies on a pre-computed Beaver's Multiplication Triple [9] (denote as MT) of the form $\langle \mathbf{c} \rangle^A = \langle \mathbf{a} \rangle^A \cdot \langle \mathbf{b} \rangle^A$. P_i computes $\langle \mathbf{e} \rangle_i^A = \langle \mathbf{x} \rangle_i^A - \langle \mathbf{a} \rangle_i^A$ and $\langle \mathbf{f} \rangle_i^A = \langle \mathbf{y} \rangle_i^A - \langle \mathbf{b} \rangle_i^A$, and recover $\mathbf{e}$ and $\mathbf{f}$ by sending to the counter-party, and let $\mathsf{Mul}(\langle \mathbf{x} \rangle_i^A, \langle \mathbf{y} \rangle_i^A) = i \cdot \mathbf{e} \times \mathbf{f} + \langle \mathbf{a} \rangle_i^A \times \mathbf{f} + \langle \mathbf{b} \rangle_i^A \times \mathbf{e} + \mathbf{c}$. The MTs can be generated offline and shared obliviously via correlated oblivious transfer extension (COT)[1] [5,20]. Details of generating MTs are given in the full version.

Yao's Garbled Circuits: Yao's protocol (aka garbled circuits (GC)) [35] empowers two secret owners evaluating an arbitrary function $\mathsf{f}(\cdot, \cdot)$ over their inputs x_0, x_1 and obtaining no more than the function's outputs z. Let $GI(z) \leftarrow \mathsf{GC}(x_0; x_1, \mathsf{f})$ denote the above procedure, where $GI(z)$ is the garbled label associated with z. The secret owners can learn the output by communicating the truth table. The point-and-permute [26] optimization allows to reuse the circuit for the same f with different inputs. Given a random in $\mathbb{Z}_{2^\ell}$, Yao's share $\langle x \rangle^Y$ and Arithmetic share $\langle x \rangle^A$ can be converted via modulo subtraction/addition inside the circuits.

4 Our Proposed Design

4.1 Architecture and Assumptions

Figure 1 depicts our architecture overview. It comprises three entities: the owners of medical time series data (aka "*hospitals*" for brevity), the *querier*, and the computational services (aka "*services*"). In our setting, the hospitals gather and archive their medical datasets independently, and a querier holds a patient's data (i.e., the query). Both of them require guaranteed privacy, while all entities perform in a *semi-honest* manner. Everyone will not deviate from the protocol but aiming to deduce the private inputs supplied by other entities. The DTW query [2] is issued by the querier who can learn and recover the encrypted results that match the query at the end. We do not constrain the roles of the querier; an individual (medical researcher) or a healthcare stakeholder (hospital) is entitled to extract knowledge about a specific disease. In a typical scenario of medical data analysis, some hospital with insufficient volume of data wishes to act as the querier and seeks assistance from other hospitals to draw a robust conclusion that is only available to herself. Considering the privacy protection, we emphasize two requirements: 1) the querier has to be distinct from the computational services, and 2) each hospital should archive the dataset independently and distributes data directly to the services. In this setting, even the querier conspires with a hospital; they cannot learn private data supplied by the others.

Our security guarantee hinges on the primary requirement that the services learn nothing about the private data they are evaluating. For this reason, it

[1] [20] suggests that the OT-based Multiplication Triples generation is faster than the Homomorphic encryption-based protocol by up to three orders of magnitude.

[2] The DTW query is the process to find the sequences similar to the query based on the DTW distance within a given threshold.

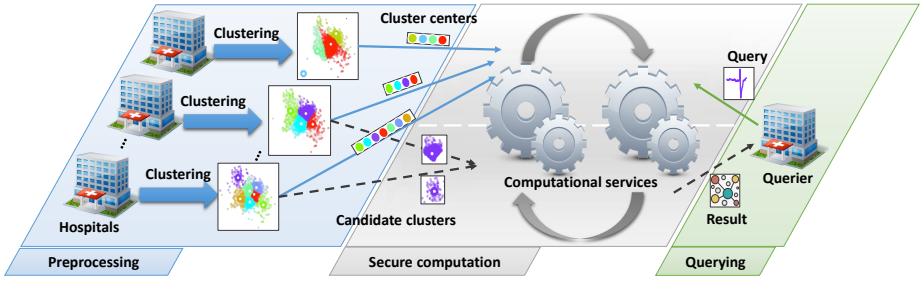

Fig. 1: System architecture

is essential to expatiate on the trust assumption of the services. They are two semi-honest but *non-colluding* services and each consists of a bunch of servers dedicating to the computation over encrypted data. In practice, they can be viewed as the full-fledged cloud services, and follow the prescribed protocols faithfully. It is reasonable since cheating would ruin their reputations. Besides, they should not be simultaneously corrupt or connive with each other. This requirement can be realized by setting up a service agreement that restricts the collusion when signing the business contracts with the two services.

Remark: The above server-aided computational model follows the rationales of prior works [27, 28]. Delegating our secure computation on two services relaxes the other parties from involving in all through. Yet it is distinct from the server-aided storage model [4] with regard to the consequences when the trust assumptions are broken. In the server-aided storage model, a logically single data owner partitions the data into shares and outsources each share in different untrusted storage services who cannot communicate mutually. Collusion leads to recover the entire dataset and incurs catastrophic consequence as the whole system is compromised directly. Whereas in our setting, collusion affects the privacy of query and query related data, since they are only data engaged in the computation and are deployed directly to the services.

4.2 Protocol Overview

Our protocol includes four phases: Preprocessing, Setup, Pruning and Analysis. The Preprocessing and Setup phases are performed offline at the local side of each party, whereas the rest two phases are undertaken online mainly between two services. The core innovation of our protocol is a secure pruning strategy via two treatments: (1) the Pruning phase for securely comparing the query with the preprocessed cluster centers to prune away the unpromising clusters whose centers are dissimilar to the query, and (2) a secure lower bounding function (the SLB function) to ensures the sequences that are not possible to be the best match being eliminated from the quadratic-time DTW computation (the SDTW function).

To support the pruning strategy, in the Preprocessing phase, each hospital clusters its local sequences into clusters, electing the centers on behalf of the corresponding clusters based on the DP algorithm. Meanwhile, the querier synthesizes an upperbound (U) and a lowerbound (L) binding the range of query

based on the LB_{Keogh} algorithm. The synthesized bounds will join the SLB computation in online phases. The intuition of preprocessing is to minimize the portion of secure computation on protected sequences, which always induces higher overheads than an equivalent calculation over the local plaintext values. Afterward, in the Setup phase, both the hospitals and the querier generate secrets of their private sequences under MPC primitives.

To realize the first treatment, the Pruning phase securely compares the query with the cluster centers only. If a center is not similar to the query, none of the sequences in the cluster it represents for would be into the querier's interest. Thus, all sequences in the cluster will be eliminated from further consideration. And if none of the centers of a hospital is similar to the query, it will be considered as an unpromising hospital, then the services will revoke the communications between them. Thereafter, the promising hospitals will be notified to supply the secrets of only candidate clusters into the Analysis phase. This treatment accelerates the overall efficiency by avoiding a considerable amount of sequences from the sequential scan in the Analysis phase. Besides, involving only the promising hospitals in the secure computation can relax other data owners, thus benefiting the case that whose medical devices possessing the data sources cannot always stay online, such as the wearable health-monitor devices and network-enabled implantable medical devices [16].

The second treatment is integrated throughout the online phases. Upon receiving the query, U, and L, whenever in the phase of comparing query with cluster centers (the Pruning phase) or scanning through all sequences in candidate clusters (the Analysis phase), the services always run the SLB function at first to abandon the unpromising sequences as early as possible. Afterward, the SDTW function is carried out between the query and eligible sequences which are closer enough according to the output of the SLB function. This treatment is economical since SLB involves fewer operations and can be smoothly proceeded in parallel, unlike the must-be-sequential SDTW computation. At the end of the Analysis phase, the services send the result back to the querier. Upon receiving the result, the querier locally reconstructs it yet knowing nothing else.

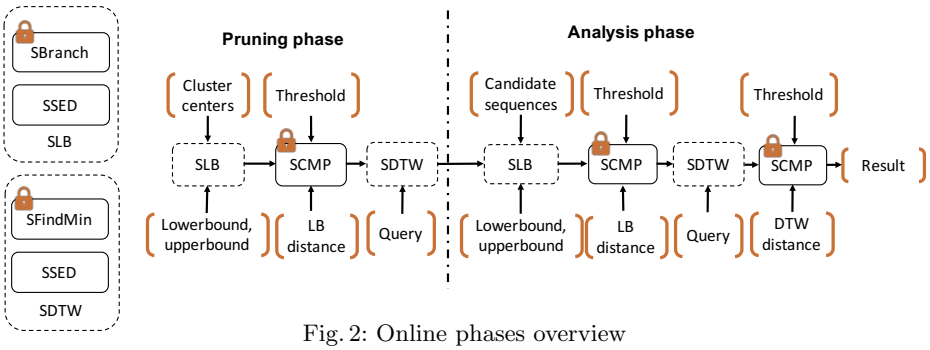

Fig. 2: Online phases overview

4.3 Design Rationale

As described above, our secure DTW-based medical analysis protocol is conducted via a hybrid approach that any time series sequences are tested by the LB distance ahead of calculating the DTW distance, defined in Eq 2 and Eq 1, respectively. We observe that both equations are performed in an iterative way comprising a series of aggregations. Each aggregation can be divided into two subtasks: (1) computing the secure squared Euclidean distance (SED), and (2) comparison among two items (for LB distance) or three items (for DTW distance). The former consists of only additions and multiplications, and the latter can be achieved by boolean operations. To tackle the above subtasks in a privacy-preserving manner, we choose Arithmetic Sharing for computing secure SED distance and Yao's GC for comparison regarding the reasons below.

1) Using Yao's GC for all aggregations requires a monolithic circuit solving a linear system as follows. On inputs dim-dimensional vectors $\mathbf{x}_i$ of query X and $\mathbf{y}_j$ of candidate Y, the circuit sequentially computes $(\mathbf{x}_i - \mathbf{y}_j)^2$ along with a comparison, and sums up the result of each iteration as the input of the next iteration. The size of the circuit relies on both dim and the sequence lengths. Putting aside the design difficulty in practice, evaluation on such a large circuit is quit time-consuming. Meanwhile, the circuit cannot be reused on different queries with various lengths due to the dependence between the circuit size and the sequence lengths. We further observe that dim is relatively small in real applications and the comparison involves few items. Thus, we apply GC only for comparison. With optimization [26], we can build the circuit once at the setup phase and used in several epochs . Note that prior work [20] shows that for an atomic comparison circuit on an integer (i.e., 32-bit values) with long-term security parameter, Yao's GC introduces less query time and bandwidth because of its constant round of interaction.

2) To avoid the overhead from crypto-operations and considerable data transfer, we choose Arithmetic Sharing to protect the data as random shares in $\mathbb{Z}_{2^\ell}$. For $\ell = 32$-bit operands, it depicts the asymptotic communication [20]. As a result, the online phase is much faster as only a few data (less than 1MB for 10^4 data points) are transferred beyond the private shares, particularly in our cross-institute scenario.

Given the rationale, the Setup phase performs offline to generate the shares of private data, and prepare MTs assisting Arithmetic multiplication. As depicted in Figure 2, the online phases involves three distance functions SSED, SLB and SDTW, and three GC-based gadgets SBranch, SFindMin, and SCMP. The Pruning phase securely measures the LB and DTW distances between the centers and query via the SLB and the SDTW functions, respectively. Similarly, the Analysis phase computes LB and DTW between candidate sequences and query. All distances are compared with threshold via the SCMP gadget. Both the SLB and SDTW functions resort the SSED function to compute the SED distance. While the SLB resorts the SBranch gadget to determine the LB distance between sequence and U, L based on their rank, the SDTW uses SFindMin to find the minimum cumulative distance.

Gadget $\mathsf{SBranch}(\langle \mathbf{m_1} \rangle^A, \langle \mathbf{m_2} \rangle^A, \langle c_1 \rangle^A, \langle c_2 \rangle^A, \omega)$:

- $S_1 \rightarrow S_0$ sends a pre-built circuit of function f along with its truth table.
- S_0 evaluates $GI(z) \leftarrow \mathsf{Y2A}(\mathsf{A2Y}(\mathsf{GC}(\omega, \langle \mathbf{m_1} \rangle_1^A, \langle \mathbf{m_2} \rangle_1^A, \langle c_1 \rangle_1^A, \langle c_2 \rangle_1^A; \langle \mathbf{m_1} \rangle_0^A, \langle \mathbf{m_2} \rangle_0^A, \langle c_1 \rangle_0^A, \langle c_2 \rangle_0^A, \mathsf{f})))$, where f performs the following:
  1. 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;
  2. compares $\mathbf{m_1}$ and $\mathbf{m_2}$, then sets a bit b that $b = 0$ indicates $\|\mathbf{m_1}\| < \|\mathbf{m_2}\|$ and $b = 1$ indicates $\|\mathbf{m_1}\| \geq \|\mathbf{m_2}\|$;
  3. if $b = 0$, sets $z = \langle dist \rangle_0^A = (c_1 - \omega) \bmod 2^\ell$ so that $\langle dist \rangle_1^A = \omega$;
  4. if $b = 1$, sets $z = \langle dist \rangle_0^A = (c_2 - \omega) \bmod 2^\ell$ so that $\langle dist \rangle_1^A = \omega$;
  5. outputs $GI(z)$ to the evaluator S_0.
- S_0 decodes according to the truth table to get his share as $\langle dist \rangle_0^A = z$.
- S_1 sets his share as $\langle dist \rangle_1^A = \omega$

Fig. 3: SBranch gadget.

4.4 Cryptographic Gadgets

Secure Branching Gadget: A branching function is that one of the two values c_1 or c_2 could be assigned to the output c depending on the rank of input messages m_1 and m_2; that is, **if** $(m_1 > m_2)$ **then** $c \leftarrow c_1$ **else** $c \leftarrow c_2$. Let c_1 denote $(\mathbf{y} - \mathbf{u})^2$ and c_2 denote $(\mathbf{l} - \mathbf{y})^2$. Eq 2 ($LB_{Keogh}$) can be viewed as two branching functions running in parallel, and formulated as: (1) **if** $(\mathbf{y} > \mathbf{u})$ **then** $dist \leftarrow c_1$ **else** $dist \leftarrow 0$; and (2) **if** $(\mathbf{l} > \mathbf{y})$ **then** $dist \leftarrow c_2$ **else** $dist \leftarrow 0$. Along with the implicit condition, i.e., $\forall \mathbf{l}, \mathbf{u}$ that $(\mathbf{l} < \mathbf{u})$, this variant is correct as it suggests three conditions $(\mathbf{y} > \mathbf{u} \&\& \mathbf{y} > \mathbf{l})$, $(\mathbf{y} < \mathbf{u} \&\& \mathbf{y} > \mathbf{l})$, and $(\mathbf{y} < \mathbf{u} \&\& \mathbf{y} < \mathbf{l})$ corresponding to the result that $dist$ could be c_1, 0, or c_2, respectively. A naive approach solving this branching function is to determine the rank of $\mathbf{y}$ and $\mathbf{u}$ (or $\mathbf{l}$) via GC, which compares its inputs, and outputs a bit indicating their rank, so that S_0 and S_1 can choose the result from c_1 and c_2. However, leaking the rank of every vector in Y and $\{U, L\}$ allows an adversary to estimate a tight range of query X through adaptive testings with a set of evenly incremented Y. Thus, it is desired to devise a secure branching scheme solving two branching functions at the same time, and obliviously assigns the shares of distance to S_0 and S_1 according to the rank of two inputs.[3] In addition, the outputs of the scheme should be well masked. Even after S_0 and S_1 adding the two outputs up, they cannot deduce the conditions based on the shares they have received. We now describe our proposed secure branching gadget SBranch. Given S_1 as the *generator* and S_0 as the *evaluator*, the shared messages $\mathbf{m_1}$, $\mathbf{m_2}$, the shares of the values to be selected c_1, c_2, a pre-build circuit, and $\omega \in \mathbb{Z}_{2^\ell}$. The SBranch gadget is detailed in Figure 3. It hides the rank of $\mathbf{m_1}$ and $\mathbf{m_2}$ because S_* always obtains his share as a randomly generated value distributed in $\mathbb{Z}_{2^\ell}$.

Remark: Prior work [27] constructs a branching scheme via GC to test if the input lies in a constant interval. Their scheme cannot directly apply to solve our branching assignments in LB_{Keogh}, where the conditions rely on the values of variables $\mathbf{u}$ and $\mathbf{l}$. Another work [8] represents a linear branching program as a

[3] Another way is building a monolithic circuit to solve a decision tree. This is not under our consideration, since it leads higher latency.

decision tree, where each input is encrypted by homomorphic cryptosystem and comparison is performed via GC, thus introducing heavy crypto-operations.

Gadget $\mathsf{SFindMin}(\langle a\rangle^A, \langle b\rangle^A, \langle c\rangle^A)$:
- S_1 generates $r \in_R \mathbb{Z}_{2^\ell}$ at random.
- $S_1 \rightarrow S_0$ sends a pre-built circuit of function $\mathsf{f}(\cdot)$ along with its truth table.
- S_0 evaluates $GI(z) \leftarrow \mathsf{Y2A}(\mathsf{A2Y}(\mathsf{GC}(r, \langle a\rangle_1^A, \langle b\rangle_1^A, \langle c\rangle_1^A;\ \langle a\rangle_0^A, \langle b\rangle_0^A, \langle c\rangle_0^A, \mathsf{f})))$, where f performs the followings:
  1. switches $\langle a\rangle^A$, $\langle b\rangle^A$ and $\langle c\rangle^A$ to a, b and c via modular additions;
  2. compares a, b and c to find the minimum value D_{min};
  3. switches D_{min} back to $\langle D_{min}\rangle^A$ by performing modular subtraction, where $\langle D_{min}\rangle_1^A = r$ and $z = \langle D_{min}\rangle_0^A = (D_{min} - r) \bmod 2^\ell$;
  4. outputs $GI(z)$ to the evaluator S_0.
- S_0 decodes according to the truth table to get his share $\langle D_{min}\rangle_0^A = z$.
- S_1 sets his share $\langle D_{min}\rangle_1^A = r$.

Fig. 4: SFindMin gadget.

Secure Find Minimum Gadget: Gadget SFindMin is used to find the minimum value among three given shares, and generate new shares of the minimum value. In our protocol, SFindMin is invoked by the SDTW function. It chooses the minimum cumulative distance $\langle D_{min}\rangle^A$ and re-generates new shares. This operation hides the index of the D_{min}, because revealing the index can leak the optimal warping path of X and Y, and further let the adversary estimate the range of X (or Y). Given S_1 as the *generator* and S_0 as the *evaluator*, a pre-built circuit, the SFindMin gadget is detailed in Figure 4. Let S_1 generate a random $r \in \mathbb{Z}_{2^\ell}$ as his share $\langle D_{min}\rangle_1^A$, gadget SFindMin outputs $\langle D_{min}\rangle_0^A$ to S_0.

Gadget $\mathsf{SCMP}(\langle a\rangle^A, \langle b\rangle^A)$:
- S_1 generates $r \in_R \mathbb{Z}_{2^\ell}$ at random.
- $S_1 \rightarrow S_0$ sends a pre-built circuit of function f along with its truth table.
- S_0 evaluates $GI(z) \leftarrow \mathsf{Y2A}(\mathsf{A2Y}(\mathsf{GC}(\langle a\rangle_1^A, \langle b\rangle_1^A; \langle a\rangle_0^A, \langle b\rangle_0^A, \mathsf{f})))$, where f switches $\langle a\rangle^A$ and $\langle b\rangle^A$ to a and b via modular additions, and compares a and b. Then sets a bit z that if $z = 0, a < b$ and if $z = 1, a \geq b$. Then sets $z = (a - r) \bmod 2^\ell$ if $a < b$, and $z = (b - r) \bmod 2^\ell$ if $a \geq b$.
- S_0 decodes $GI(z)$ according to the truth table to obtain z.

Fig. 5: SCMP gadget.

Secure Comparison Gadget: Gadget SCMP is used to compare two given shares and outputs a bit indicating their rank via GC. In our protocol, it determines whether a specific distance is within a matching threshold. Given S_1 as the *generator* and S_0 as the *evaluator*, a pre-build circuit, the gadget SCMP compares its two inputs and output a bit indicating the rank. The SCMP gadget is detailed in Figure 5. It can be realized in two versions achieving different security strengths and performance. Version 1 reveals the rank of LB/DTW distances and threshold when comparing the cluster centers, since this information is not directly related to the private inputs and protocol results. Version 2 in color

compares DTW distances of candidate sequences and threshold, and outputs the shares of rank securely.

Function $\mathsf{SSED}(\langle \mathbf{x} \rangle^A, \langle \mathbf{y} \rangle^A, \langle \mathbf{x}^2 \rangle^A, \langle \mathbf{y}^2 \rangle^A, MT)$:
• S_* sets $\langle dist \rangle_*^A = \langle \mathbf{x}^2 \rangle_*^A + \langle \mathbf{y}^2 \rangle_*^A - 2(\mathsf{Mul}^A(\langle \mathbf{x} \rangle_*^A, \langle \mathbf{y} \rangle_*^A))$, where $* \in \{0, 1\}$.

Fig. 6: SSED function.

4.5 Distance Functions

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 function SSED to attain $\langle dist \rangle_0^A$ and $\langle dist \rangle_1^A$ as shown in Figure 6.

Algorithm 1 Secure LB_{Keogh} function

function $\mathsf{SLB}(\langle U \rangle^A, \langle U^2 \rangle^A, \langle L \rangle^A, \langle L^2 \rangle^A, \langle Y \rangle^A, \langle Y^2 \rangle^A, MTs)$
  S_* initializes $\langle LB \rangle_*^A$ and $r \leftarrow 0$ for $* \in \{0, 1\}$.
  for $i \in [1, |X|]$ **do**
    S_* initializes $\langle dist_1 \rangle_*^A$, $\langle dist_2 \rangle_*^A$, $\langle c_1 \rangle_*^A$ and $\langle c_2 \rangle_*^A$.
    S_1 generates $\omega_i^1, \omega_i^2 \in \mathbb{Z}_{2^\ell}$ at random.
    S_* runs to get $\langle c_1 \rangle_*^A \leftarrow \mathsf{SSED}(\langle \mathbf{u}_i \rangle^A, \langle \mathbf{y}_i \rangle^A, \langle \mathbf{y}_i^2 \rangle^A, \langle \mathbf{u}_i^2 \rangle^A, MT)$
    S_* runs to get $\langle c_2 \rangle_*^A \leftarrow \mathsf{SSED}(\langle \mathbf{l}_i \rangle^A, \langle \mathbf{y}_i \rangle^A, \langle \mathbf{y}_i^2 \rangle^A, \langle \mathbf{l}_i^2 \rangle^A, MT)$.
    S_* runs to get $\langle dist_1 \rangle_*^A \leftarrow \mathsf{SBranch}(\langle \mathbf{u} \rangle^A, \langle \mathbf{y} \rangle^A, \langle c_1 \rangle^A, r, \omega_i^1)$.
    S_* runs to get $\langle dist_2 \rangle_*^A \leftarrow \mathsf{SBranch}(\langle \mathbf{y} \rangle^A, \langle \mathbf{l} \rangle^A, \langle c_2 \rangle^A, r, \omega_i^2)$.
    S_* sets $\langle dist \rangle_*^A = \langle dist_1 \rangle_*^A + \langle dist_2 \rangle_*^A$, and $\langle LB \rangle_*^A = \langle LB \rangle_*^A + \langle dist \rangle_*^A$.
  end for
  S_* gets $\langle LB \rangle_*^A$.
end function

Secure LB_{Keogh} Function: Algorithm 1 shows the SLB function running between sequence Y and two synthesized sequences U and L. It iteratively sums up $|X|$ numbers of secure SED distances indicating how far Y falls out of the range of X bound by U and L. Let $\langle Y \rangle^A$, $\langle Y^2 \rangle^A$, $\langle U \rangle^A$, $\langle U^2 \rangle^A$, $\langle L \rangle^A$, $\langle L^2 \rangle^A$ denote the shares of Y, U, and L, and the shares of their squared values, respectively. On inputs these shares, and MTs, the SLB function outputs $\langle LB \rangle_*^A$ in private. In detail, once it is launched, S_* initialize variable $r = 0$ as one of the distances awaiting to be selected by the SBranch gadget, where $* \in \{0, 1\}$. In each iteration, S_1 generates randomnesses $\omega_i^1, \omega_i^2 \in \mathbb{Z}_{2^\ell}$. Afterwards, S_* invoke the SSED function to compute the shares of $dist(\langle \mathbf{y}_i \rangle^A, \langle \mathbf{u}_i \rangle^A)$ as $\langle c_1 \rangle_*^A$, and the shares of $dist(\langle \mathbf{y}_i \rangle^A, \langle \mathbf{l}_i \rangle^A)$ as $\langle c_2 \rangle_*^A$. They then invoke SBranch to assign one of the values $\langle c_1 \rangle_*^A$ (or $\langle c_2 \rangle_*^A$) and r to $\langle dist_1 \rangle_*^A$ (or $\langle dist_2 \rangle_*^A$), according to the rank of $\mathbf{y}$ and $\mathbf{u}$ (or $\mathbf{l}$). The output distance is masked with ω_i^1 (or ω_i^2). In other words, the value of $\langle dist_1 \rangle^A$ could be c_1 or 0, while the value of $\langle dist_2 \rangle^A$ could be c_2 or 0. This treatment is correct, since after S_0 and S_1 adding them up, the value of $\langle dist \rangle^A$ could be c_1, c_2 or 0, yet not $c_1 + c_2$, because of the implicit condition $\mathbf{u} > \mathbf{l}$ excludes this case. At the end of each iteration, S_* add $\langle dist \rangle_*^A$ on the

$\langle LB \rangle_*^A$ attained in the previous iteration. Ultimately, they get the final result of secure LB distance between Y and $\{U, L\}$.

Algorithm 2 Secure DTW function:

function SDTW($\langle X \rangle^A, \langle X^2 \rangle^A, \langle Y \rangle^A, \langle Y^2 \rangle^A, MTs$)
  S_* initializes $\langle DTW \rangle_*^A$ and two arrays $cost_*$ and $cost_{prev,*}$ with numeric infinity (INF), that each has $(2cr + 1)$ numbers of cells for $* \in \{0, 1\}$.
  for $i \in [0, |X| - 1]$ **do**
    S_* initializes $k \leftarrow max(0, cr - i)$.
    for $j \in [\ \max(0, i - cr),\ \min(|X| - 1, i + cr)]$ **do**
      S_* initializes $\langle a \rangle_*^A$, $\langle b \rangle_*^A$, $\langle c \rangle_*^A$ with INF, $\langle dist \rangle_*^A$, and $\langle D \rangle_*^A$.
      If $i = 0$ && $j = 0$, S_* jointly run to get $\langle dist \rangle_*^A \leftarrow$ SSED($\langle \mathbf{x}_0 \rangle^A, \langle \mathbf{y}_0 \rangle^A, \langle \mathbf{x}_0^2 \rangle^A, \langle \mathbf{y}_0^2 \rangle^A, MT$), and stores each share at $cost_*[k]$. S_* sets k ++, and continues to next iteration.
      If $j \geq 1$ && $k \geq 1$, S_* sets $\langle b \rangle_*^A \leftarrow cost_*[k - 1]$; else INF.
      If $i \geq 1$ && $k + 1 \leq 2 * cr$, S_* sets $\langle a \rangle_*^A \leftarrow cost_{perv,*}[k + 1]$; else INF.
      If $i \geq 1$ && $j \geq 1$, S_* sets $\langle c \rangle_*^A \leftarrow cost_{prev,*}[k]$; else INF.
      S_* run $\langle D \rangle_*^A \leftarrow$ SSED($\langle \mathbf{x}_i \rangle^A, \langle \mathbf{y}_j \rangle^A, \langle \mathbf{x}_i^2 \rangle^A, \langle \mathbf{y}_j^2 \rangle^A, MT$) + SFindMin ($\langle a \rangle^A, \langle b \rangle^A, \langle c \rangle^A$), and stores each share at $cost_*[k]$, and then sets k ++.
    end for
    S_* copies $cost_*$ to $cost_{prev,*}$, and cleans $cost_*$.
  end for
  S_* gets $\langle DTW \rangle_*^A \leftarrow cost_{prev,*}[cr]$.
end function

Secure DTW Function: The SDTW function, as Algorithm 2 illustrates, is the core building block as it measures whether a sequence Y matches the given query X based on DTW. On inputs $\langle Y \rangle^A$, $\langle Y^2 \rangle^A$, $\langle X \rangle^A$, $\langle X^2 \rangle^A$ and MTs, it returns $\langle DTW \rangle^A$. Because the parameter cr of global constraint is a data-independent constant, we assume it is known by S_0 and S_1 as a system parameter. We briefly sketch the philosophy of realizing the SDTW function. We maintain a pair of arrays $cost$ and $cost_{prev}$ with $2cr + 1$ cells to record the cumulative distances between X_i, X_{i+1} and Y. For each i, the arrays act as vertical bars moving from left to right and bottom-up one cell each iteration. Let us consider how to calculate one distance $D(X_i, Y_j)$ only. Assume cell $cost[k]$ store $D(X_i, Y_j)$, where $k \in [max(0, cr - i), 2cr + 1]$. We first calculate $dist(\mathbf{x_i}, \mathbf{y_j})$. We then check whether the awaiting calculated $D(X_i, Y_j)$ is located at the edge of the sliding window formed by the global constraint, i.e., at $cost_*[0]$ or $cost_*[2cr + 1]$. If not, we find the minimum value among its adjacent cumulative distances, i.e., $D(X_i, Y_{j-1})$, $D(X_{i-1}, Y_j)$ and $D(X_{i-1}, Y_{j-1})$ located at cells $cost[k - 1]$, $cost_{prev}[k + 1]$ and $cost_{prev}[k]$. Otherwise, we retrieve the minimum among the existences of the above three values. We add the above results up as $D(X_i, Y_j)$, and store into cell $cost[k]$. At the end, the DTW is located at $cost_{prev}[cr]$. Following the above methodology, S_0 and S_1 iteratively calculate $\langle dist \rangle_*^A$ via the SSED function, find the minimum $\langle D_{min} \rangle_*^A$ among $\langle a \rangle_*^A$, $\langle b \rangle_*^A$ and $\langle c \rangle_*^A$ via the SFindMin gadget, and sum them up as $\langle D \rangle_*^A$. Ultimately, S_0 and S_1 obtain $\langle DTW \rangle_*^A$ at $cost_{prev,*}^A[cr]$.

Protocol Φ:

$\mathsf{Setup}(X, U, L, \{\mathbf{C}_1, ..., \mathbf{C}_m\}, \Delta)$:

1: S_0 and S_1 interactively generate M sets of MTs. S_1 generates three circuits for the $\mathsf{SBranch}$, $\mathsf{SFindMin}$ and SCMP gadgets.
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_α 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 indices $r_\beta \in \{r_1, ..., r_k\}$ randomly for each cluster, where $\beta \in [1, k]$, and distributes them to S_0 and S_1.

$\mathsf{Pruning}(\langle U \rangle^A, \langle U^2 \rangle^A, \langle L \rangle^A, \langle L^2 \rangle^A, \langle X \rangle^A, \langle X^2 \rangle^A, \langle \mathfrak{C}^\beta \rangle^A, \langle (\mathfrak{C}^\beta)^2 \rangle^A, \langle \Delta \rangle^A)$:

1: S_0 initializes an array $\mathbf{Arr}_\alpha^C$.
2: **for** each cluster center $\mathfrak{C}^\beta \in \mathbf{C}_\alpha$ in each H_α **do**
3: S_* run $\langle LB_{\mathfrak{C}} \rangle_*^A \leftarrow \mathsf{SLB}(\langle U \rangle^A, \langle U^2 \rangle^A, \langle L \rangle^A, \langle L^2 \rangle^A, \langle \mathfrak{C}^\beta \rangle^A, \langle (\mathfrak{C}^\beta)^2 \rangle^A, MTs)$ for $* \in \{0, 1\}$.
4: If $\mathsf{SCMP}(\langle LB_{\mathfrak{C}} \rangle^A, \langle \Delta \rangle^A) == 0$, S_* run $\langle DTW_{\mathfrak{C}} \rangle_*^A \leftarrow \mathsf{SDTW}(\langle X \rangle^A, \langle X^2 \rangle^A, \langle \mathfrak{C}^\beta \rangle^A, \langle (\mathfrak{C}^\beta)^2 \rangle^A, MTs)$. Else, continue to next.
5: If $\mathsf{SCMP}(\langle DTW_{\mathfrak{C}} \rangle^A, \langle \Delta \rangle^A) == 0$, S_0 adds index r_β of cluster C^β to $\mathbf{Arr}_\alpha^C$. Else, continue to next.
6: **end for**
7: If $\mathbf{Arr}_\alpha^C = \perp$, S_* close the connections with H_α. Else, S_0 sends $\mathbf{Arr}_\alpha^C$ to H_α.
8: H_α distributes $\langle \bar{Y} \rangle^A$ of each $\bar{Y}$ in all candidate clusters $\bar{C}^\beta$ to S_0 and S_1.

$\mathsf{Analysis}(\langle U \rangle^A, \langle U^2 \rangle^A, \langle L \rangle^A, \langle L^2 \rangle^A, \langle X \rangle^A, \langle X^2 \rangle^A, \langle \bar{Y} \rangle^A, \langle \bar{Y}^2 \rangle^A, \langle \Delta \rangle^A)$:

1: S_* initialize arrays $\mathbf{Arr}_{\alpha,*}$.
2: **for** each sequence $\bar{Y}$ in all candidate clusters $\bar{C}^\beta$ **do**
3: S_* run $\langle LB \rangle_*^A \leftarrow \mathsf{SLB}(\langle U \rangle^A, \langle U^2 \rangle^A, \langle L \rangle^A, \langle L^2 \rangle^A, \langle \bar{Y} \rangle^A, \langle \bar{Y}^2 \rangle^A, MTs)$.
4: If $\mathsf{SCMP}(\langle LB \rangle^A, \langle \Delta \rangle^A) == 0$, S_* run $\langle DTW \rangle_*^A \leftarrow \mathsf{SDTW}(\langle X \rangle^A, \langle X^2 \rangle^A, \langle \bar{Y} \rangle^A, \langle \bar{Y}^2 \rangle^A, MTs)$. Else, continue to next.
5: S_* tun to get $\langle res \rangle_*^A \leftarrow \mathsf{SCMP}(\langle DTW \rangle^A, \langle \Delta \rangle^A)$and store to $\mathbf{Arr}_{\alpha,*}$.
6: **end for**
7: S_* sends $\mathbf{Arr}_{\alpha,*}$ to Q.
8: Q runs reconstruction $\mathsf{Rec}^A(res)$ and remove 0 to get each DTW in $\mathbf{Arr}_\alpha$.

Fig. 7: Secure DTW-based medical time series analysis protocol

4.6 Secure DTW-based Medical Analysis Protocol

Figure 7 describes our protocol Φ that modularly composes the above crypto-gadgets and atomic functions.

Setup Phase: Consider each hospital $H_\alpha \in \{H_1, ..., H_m\}$ holds an n-sequences dataset $\mathbf{Y}^\alpha = \{Y^1, ..., Y^n\}$. Each hospital pre-partitions the local dataset as k clusters $\mathbf{C}_1 = \{C^1, ..., C^k\}$ represented by centers $\mathfrak{C}^1, ..., \mathfrak{C}^k$. Given pre-computed synthesized U and L binding the query X, and the matching threshold Δ held by querier Q. Once the Setup phase begins, the querier and all hospitals run in parallel to generate the Arithmetic shares of their private data, and distribute the shares to S_0 and S_1. To assist multiplication over shares, S_0 and S_1 interactively derive dim-dimensional MT. $(\langle \mathbf{c} \rangle^A, \langle \mathbf{a} \rangle^A, \langle \mathbf{b} \rangle^A)$ that $\mathbf{c} = \mathbf{a} \times \mathbf{b}$. Note that single multiplication operation requires one MT. Apart from MTs, S_1 generates three circuits for the $\mathsf{SBranch}$, $\mathsf{SFindMin}$ and SCMP gadgets, respectively.

Secure Pruning Phase: The Pruning phase runs in parallel between each pair of instances of S_0 and S_1 connecting with each H_α. Without loss of generality, we discuss one pair of instances with H_α as an example. S_0 initializes a dynamic array $\mathbf{Arr}_\alpha^C$ used to store the dummy index r_β of the potential clusters C^β. S_0 and S_1 jointly run the SLB function to get $\langle LB_{\mathfrak{C}} \rangle_*^A$ between $\mathfrak{C}^\beta$ and $\{U, L\}$, where $* \in \{0, 1\}$. They then invoke the SCMP gadget to check if the resulted $\langle LB_{\mathfrak{C}} \rangle_*^A$ is less than $\langle \Delta \rangle^A$. If so, they run the SDTW function to get $\langle DTW_{\mathfrak{C}} \rangle_*^A$ and check whether it is less than $\langle \Delta \rangle^A$. If it is, S_0 adds the dummy index r_β of the currently processed $\mathfrak{C}^\beta$ to $\mathbf{Arr}_\alpha^C$. After checking all $\mathfrak{C}^\beta$, if $\mathbf{Arr}_\alpha^C$ is not empty, S_0 sends it to H_α, and H_α sends back the shares of each sequence $\bar{Y}$ in candidate clusters $\bar{C}^\beta$ corresponding to r_β. Otherwise, S_0 and S_1 revoke the connection with H_α who does not need to stay online anymore.

Secure DTW-based Analysis Phase: The Anaysis phase runs in parallel between S_0 and S_1 with each candidate hospital $\bar{H}_\alpha$. S_* initializes dynamic array $\mathbf{Arr}_{\alpha,*}$ to store the shares of resulting secure DTW distances, where $* \in \{0, 1\}$. For each candidate sequence $\bar{Y}$ (i.e., the results of previous phase), S_* runs the SLB function to get the shares of LB distance between $\bar{Y}$ and X, denoted as $\langle LB \rangle_*^A$. S_* then calls the SCMP gadget to compare $\langle LB \rangle^A$ and $\langle \Delta \rangle^A$. If $\langle LB \rangle_*^A$ is within the threshold, they run SDTW to get $\langle DTW \rangle_*^A$. Services then input $\langle DTW \rangle^A$ and $\langle \Delta \rangle^A$ to the SCMP gadget which outputs the shares of the smaller one, and store to $\mathbf{Arr}_{\alpha,*}$. Ultimately, S_0 and S_1 send $\mathbf{Arr}_{\alpha,*}$ to Q who then reconstructs the results, excludes the values identical to threshold, and get DTWs indicating how likely an individual will have a specific disease.

Remark of Complexity: Suppose sequence length $|X|$, global constraint cr, pruning ratio σ of the SLB function, and pruning ratio μ of the Pruning phase. Given all cluster centers $\mathfrak{C}^{\alpha,\beta}$, all dataset $\mathbf{Y}^\alpha$ with n sequences, where $\alpha \in [1, m], \beta \in [1, k]$. Computing one SLB function requires $2|X|$ calls of the SSED function and the SBranch gadget. Computing one SDTW function runs $|X| \cdot (2cr + 1) - cr \cdot (cr + 1)$ (denoted as θ_{sdtw}) times the SSED function and the SFindMin gadget. The number of calls of atomic functions is summarized in Table 1.

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

Table 1: Number of calls of atomic functions

4.7 Security Guarantee

For our multi-instance and concurrent-execution protocol Φ, we define security following the *Universally Composable* (UC) security framework [17], where under a general protocol composition operation (*universal composition*), the security of our protocol is preserved. Consider engaged parties a querier Q, hospitals $H_1, ..., H_m$ and two non-colluding services S_0, S_1. Suppose a semi-honest *admissible adversary* $\mathcal{A}$ who can corrupt the querier, any subset of hospitals, as well as one of the two non-colluding services at most, i.e., if S_0 is compromised by $\mathcal{A}$, S_1 acts honestly; vice versa. Yet we do not restrict the collusions among the

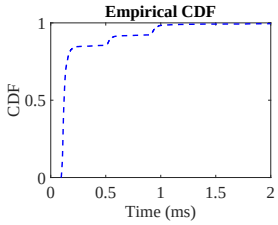

Fig. 8: SSED unit time.

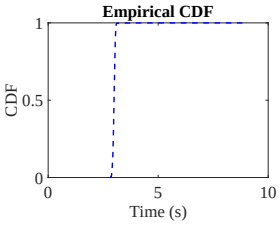

Fig. 9: SLB unit time.

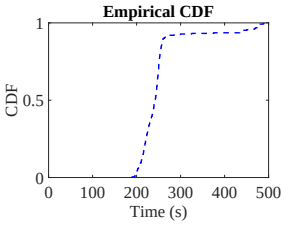

Fig. 10: SDTW unit time.

rest parties. Since our protocol directly follows the security of the Arithmetic sharing [6] and GC [35], and all medical sequences, MTs, and intermediate results are well protected as randomly generated shares in the ring $\mathbb{Z}_{2^\ell}$, 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 parties are their inputs and outputs from the protocol yet nothing about the data of the remaining honest parties. Due to space constraints, formal security proof is given in the full version of this paper.

5 Performance Evaluation

5.1 Implementation and Setup

We implement a prototype of our secure DTW-based medical analysis system in Java. For the choice of OT and GC, we utilize FlexSC [34], i.e., a Java-based toolkit implementing extended OTs [5] and optimizations of GC. Regarding Arithmetic Sharing, we set the size of the ring as 2^{31} to fit in the Java primitive type int rather than BigInteger, resulting in acceleration on modulo addition and multiplication. We deploy this prototype on 4 Amazon EC2 c5.4xlarge instances running Ubuntu 16.04 LTS with 3.0GHz Intel Xeon Platinum processor (16 vCPUs), 32GB RAM, and 10Gpbs virtual NIC each; performing two services, a hospital and a querier. The reported measurements make use of a 256Hz ECG dataset, derived from UCR Time Series Datasets at [2] (a real-world dataset from PhysioBank [1]). Our dataset contains 15K sequences and queries with length 128, formed with single dimensional vectors. The dataset is stored at in-memory Redis(v3.0.6) database as key-value pairs. We apply global constraint $cr = 0.05 * 128 = 7$ to LB and DTW [29].

To improve runtime performance, we use the in-memory Redis database to catch the intermediate results, such as the all-pair distance matrix used for DP clustering. Besides, we store a bunch of randomly generated MTs in files (300 triples each file). Once required, the services will randomly select files to retrieve a set of MTs upon the demand of computation, and then delete the files to ensure the randomization of MTs. Besides, we modify FlexSC slightly to enable concurrent processing of the SLB function.

5.2 Evaluation

Cryptographic Gadgets: The performance of crypto-gadgets is summarized in Table 7 regarding the running time on the local network, the bandwidth and the

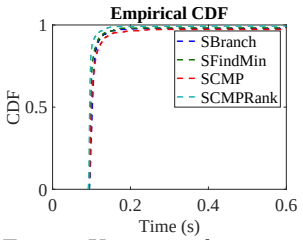

Fig. 11: Unit time for crypto-gadgets.

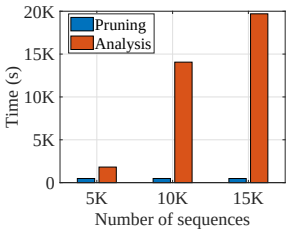

Fig. 12: Overall time for online phases.

Threshold	# clusters		
	10	20	30
10511173	96.3%	93.2%	91.0%
18848334	71.6%	69.5%	68.2%
22834481	50.0%	43.9%	41.5%

Table 2: Pruning ratio of Pruning phase

number of AND gates.All three measures are increased linearly to the number of inputs, while the time fluctuates slightly. Besides, the empirical cumulative distribution function (CDF), as shown in Figure 11, sheds light on the distribution of the time on running individual gadget. We grab 1K executions and all of four gadgets can be done within 0.4 seconds.

	SSED		SLB		SDTW	
# calls	Time	Comm.	Time	Comm.	Time	Comm.
10000	1.88s	0.30MB	18462.5s	108.7GB	2167591.4s	731.9GB
20000	3.30s	0.61MB	44609.6s	217.3GB	5221399.3s	1463.9GB
30000	4.59s	0.91MB	79900.5s	325.9GB	7551764.1s	2195.8GB

Table 3: Performance of distance functions

Distance Functions: We benchmark the execution of distance functions SSED, SLB, and SDTW. The time and bandwidth ascend linearly for all of them in line with the number of calls as Table 3 illustrates. In addition, we grasp 5K unit execution times of SSED, SLB, and SDTW to form their distributions by empirical CDF. As exhibited in Figure 8, the unit call of SSED can be finished within 1.5ms. Likewise, Figure 9 depicts the unit time of SLB, where 4s is sufficient to a single execution. We further estimate the theoretical unit time of SLB, combining 128*2 calls of SSED and SBranch. Given the unit time of SBranch as 0.055s and SSED as 1.5ms, the theoretical unit time is 15s. The retrenchment of time consumed in practice is contributed by the concurrent implementation, where every distance between each two data vectors can be calculated by independent thread in parallel. Similarly, Figure 10 illustrates the unit time of SDTW, which is larger than 200s. This confirms the reasonableness of our pruning strategy. Yet we remark this experiment result does not beyond our expectation, because running SDTW function over 10K sequences involves $1.92 * 10^7$ calls of SSED and SFindMin.

Online Phases: We then turn our attention from runtime overhead of atomic operations to the workload of each party in each phase of our protocol. The overall time of online phases is evaluated in chunks of 5K sequences with threshold 22834481. Each set of data is randomly selected from our dataset and partitioned in 10 clusters. We report the number of sequences similar to the query as 8 for 5K data, 59 for 10K data, and 87 for 15K data. As Figure 12 depicts, followed by the growth of dataset, the run-time of Pruning phase remains a flat trend as its computational overhead depends on the number of clusters. In contrast, the

time of Analysis phase ascends sharply yet does not form linear increase, since its workload depends on two aspects: (1) the sequential test under the SLB function over all candidate sequences submitted to Analysis phase, and (2) the quadratic time SDTW computation over resulting sequences from the SLB function.

The effectiveness of applying Pruning phase is confirmed via pruning ratios with 15K sequences classified in 10, 20 and 30 clusters and a group of thresholds randomly extracted from DTW distances. The ratio is qualitative as the number of excluded sequences divides the size of dataset. As Table 2 exhibits, the ratio drops down with the increasing amount of clusters and values of threshold, yet avoiding at least 6K sequences from the sequential scan.

# MTs	Time	Comm.
10^5	610.4s	2.9GB
10^7	60885.1s	292.6GB
10^9	6052449.5s	29263.2GB

Table 4: Setup phase performance of services

# queries	Time (LB/UB)	Time (sharing)
1	2.0ms	1.3ms
100	10.6ms	10.5ms
1000	51.5ms	38.5ms

Table 5: Preprocess and setup phases performance of querier

# sequences	Time (DP)	Time (sharing)
5000	5467.6s	0.2s
10000	55558.2s	0.3s
15000	421895.4s	0.6s

Table 6: Preprocess and setup phases performance of hospital

Preprocess and Setup Phases: Table 4 shows the time and bandwidth costs of two services in the Setup phase. Both of them rise linearly with the amount of MTs. Table 5 shows the light workload of the querier, encompassing synthesizing U and L in Preprocessing phase and generating shares of queries in Setup phase. Table 6 shows the time cost of the hospital. Despite the intensive workload brought by DP [30] clustering in Preprocessing phase, the Setup phase does not aggravate its workload, as generating shares of 15K sequences can be completed within 1s.

SBranch				SFindMin			
# inputs	Time	Comm.	Gates	# inputs	Time	Comm.	Gates
10^5	605.1s	3.0GB	$859*10^5$	10^6	5533.2s	26.9GB	$764*10^6$
10^6	6190.5s	30.2GB	$859*10^6$	10^7	58915.2s	268.9GB	$764*10^7$
10^7	69387.3s	301.5GB	$859*10^7$	10^8	576590.7s	2689.6GB	$764*10^8$
SCMP				SCMP (leak rank)			
# inputs	Time	Comm.	Gates	# inputs	Time	Comm.	Gates
10^4	46.5s	0.2GB	$541*10^4$	10^4	39.5s	0.17GB	$350*10^4$
10^5	455.1s	2.1GB	$541*10^5$	10^5	389.3s	1.68GB	$350*10^5$
10^6	4652.6s	21.3GB	$541*10^6$	10^6	3846.5s	16.8GB	$350*10^6$

Table 7: Performance of gadgets.

6 Conclusion

In this paper, we propose a privacy-preserving DTW-based analysis system over distributed medical time series data. Our system constructs a scalable architecture providing dedicate computational services that allow multiple healthcare institutes to carry out secure joint medical data computation. Atop this architecture, our system enables a mixed protocol with tailored atomic functions under MPC primitives. The composed protocol empowers rich expressive power to support various functionality and strong security guarantee. To descend the query

latency, together with preprocessing, we devise a two-layer pruning strategy which reduces the portion of secure computation and excludes the unpromising sequences from a sequential scan under DTW. Comprehensive empirical validation shows the potential of our system to be deployed in practice.

Acknowledgment This work was supported by Australian Research Council Discovery and Linkage Projects (DP180103251 and LP160101766).

References

1. Physiobank atm. Online at http://physionet.org/cgi-bin/atm/ATM
2. Ucr time series classification archive. Online at https://www.cs.ucr.edu/~eamonn/time_series_data_2018/
3. 104th United States Congress: Health Insurance Portability and Accountability Act of 1996 (HIPPA). online at https://www.hhs.gov/hipaa/index.html (1996)
4. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. Proc. of CIDR (2005)
5. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: Proc. of ACM CCS (2013)
6. Atallah, M., Bykova, M., Li, J., Frikken, K., Topkara, M.: Private collaborative forecasting and benchmarking. In: Proc. of WPES (2004)
7. Baldi, P., Baronio, R., De Cristofaro, E., Gasti, P., Tsudik, G.: Countering gattaca: efficient and secure testing of fully-sequenced human genomes. In: Proc. of ACM CCS (2011)
8. Barni, M., Failla, P., Lazzeretti, R., Sadeghi, A.R., Schneider, T.: Privacy-preserving ecg classification with branching programs and neural networks. IEEE TIFS (2011)
9. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Proc. of Crypto (1991)
10. Begum, N., Ulanova, L., Wang, J., Keogh, E.: Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In: Proc. of ACM SIGKDD (2015)
11. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Proc. of KDD workshop (1994)
12. Blanton, M., Kang, A.R., Karan, S., Zola, J.: Privacy preserving analytics on distributed medical data. CoRR abs/1806.06477 (2018), http://arxiv.org/abs/1806.06477
13. Bogdanov, D., Laud, P., Randmets, J.: Domain-polymorphic language for privacy-preserving applications. In: Proc. of the ACM workshop on Language support for privacy-enhancing technologies (2013)
14. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: Proc. of ESORICS (2008)
15. Brickell, J., Porter, D.E., Shmatikov, V., Witchel, E.: Privacy-preserving remote diagnostics. In: Proc. of ACM CCS (2007)
16. Camara, C., Peris-Lopez, P., Tapiador, J.E.: Security and privacy issues in implantable medical devices: A comprehensive survey. Journal of biomedical informatics 55, 272–289 (2015)

17. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000)
18. Chen, Y., Hu, B., Keogh, E., Batista, G.E.: Dtw-d: time series semi-supervised learning from a single example. In: Proc. of ACM SIGKDD (2013)
19. Cho, H., Wu, D.J., Berger, B.: Secure genome-wide association analysis using multiparty computation. Nature Biotechnology 36(6), 547–551 (2018)
20. Demmler, D., Schneider, T., Zohner, M.: Aby-a framework for efficient mixed-protocol secure two-party computation. In: Proc. of NDSS (2015)
21. 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)
22. Huang, Y., Malka, L., Evans, D., Katz, J.: Efficient privacy-preserving biometric identification. In: Proc. of NDSS (2011)
23. Keogh, E.: Exact indexing of dynamic time warping. In: Proc. of VLDB (2002)
24. Kerschbaum, F., Schneider, T., Schröpfer, A.: Automatic protocol selection in secure two-party computations. In: Proc. of ACNS (2014)
25. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of cryptology 15(3) (2002)
26. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y., et al.: Fairplay-secure two-party computation system. In: Proc. of USENIX Security (2004)
27. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: Proc. of IEEE S&P (2017)
28. Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., Taft, N.: Privacy-preserving ridge regression on hundreds of millions of records. In: Proc. of IEEE S&P (2013)
29. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proc. of ACM SIGKDD (2012)
30. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science 344(6191), 1492–1496 (2014)
31. Salem, A., Berrang, P., Humbert, M., Backes, M.: Privacy-preserving similar patient queries for combined biomedical data. Proc. of PETS (2019)
32. Tkachenko, O., Weinert, C., Schneider, T., Hamacher, K.: Large-scale privacy-preserving statistical computations for distributed genome-wide association studies. In: Proc. of ACM AsiaCCS (2018)
33. Wang, X.S., Huang, Y., Zhao, Y., Tang, H., Wang, X., Bu, D.: Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In: Proc. of ACM CCS (2015)
34. Wang, X.: Flexsc. Online at `https://github.com/wangxiao1254/FlexSC` (2018)
35. Yao, A.C.C.: How to generate and exchange secrets. In: Proc. of IEEE FOCS (1986)
36. Yi, X., Bertino, E., Rao, F.Y., Bouguettaya, A.: Practical privacy-preserving user profile matching in social networks. In: Proc. of IEEE ICDE (2016)
37. Zheng, W., Popa, R., Gonzalez, J.E., Stoica, I.: Helen: Maliciously secure coopetitive learning for linear models. In: Proc. of IEEE S&P (2019)
38. Zheng, Y., Duan, H., Tang, X., Wang, C., Zhou, J.: Denoising in the dark: Privacy-preserving deep neural network based image denoising. IEEE TDSC (2019)
39. Zheng, Y., Duan, H., Wang, C.: Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing. IEEE TIFS 13(10), 2475–2489 (2018)
40. Zhu, H., Meng, X., Kollios, G.: Privacy preserving similarity evaluation of time series data. In: Proc. of EDBT (2014)

7 Security Analysis

We define the security following the *Universally Composable* (UC) security framework [17]. Given the protocol Φ, each instance of Φ executed by the parties runs as subroutine of multiple interactive Turing Machines (ITMs). Given the target ideal functionality $\mathcal{F}$. Suppose a polynomial-time semi-honest *admissible adversary* $\mathcal{A}$ who can corrupt querier, any subset of hospitals, as well as one of the two services 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 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 is:

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

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 function $\mathcal{G}$ for $\mathcal{A}$, in the context that Φ can have multiple instances of $\mathcal{G}$ operated concurrently. The UC theorem [17] 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 Figure 13. 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.

Theorem 1. *Let protocol Φ be the* Analysis *phase defined in Figure 7. 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 Figure 13 against a polynomial-time semi-honest admissible adversary.*

Proof. Let $\Phi^{\mathcal{G}\to\rho}$ 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 $\Phi^{\mathcal{G}\to\rho}$. 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 overall simulator $\mathcal{S}$ simulates $\mathcal{A}$ in 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 Arithmetic 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 Arithmetic shares and GC. To handle the revealed rank, $\mathcal{S}$ maintains a key-value stored dictionary T to record the relationship between the rank outputted 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, $\mathcal{S}$ then checks T to see if it has the key (distance, threshold). If the key exists, $\mathcal{S}$ forwards T[(distance, threshold)] to $\mathcal{E}$. Otherwise, upon receiving the rank outputted from $\mathcal{S}_\rho$, $\mathcal{S}$ forwards the rank to $\mathcal{E}$ and saves it as T[(distance, threshold)] = rank. To this end, we argue that the $\mathcal{E}$'s views of real and ideal world are identical.

Functionality $\mathcal{F}$: Interact with dummy querier Q, hospitals $H_1, ..., H_m$, services S_0, S_1, and an adversary $\mathcal{S}$.

Initiate Upon receiving $\mathsf{initiate}$ calls from all parties, sends the message to $\mathcal{S}$, initializes an 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, set $T[k] = \langle X\rangle^A || \langle U\rangle^A || \langle L\rangle^A || \langle \Delta\rangle^A$, or $(\mathsf{Input}, \alpha, k, \langle Y\rangle^A)$ from H_α, set $T[k] = \langle Y\rangle^A$, 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 $(\mathsf{SLB}, x, y, z, w)$, sets $T[z] = \mathcal{F}_{slb}(T[x], T[y], T[w])$ and sends $(\mathsf{SLB}, x, y, z, w)$ to S_0, S_1 and $\mathcal{S}$.

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 and $\mathcal{S}$.

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 and $\mathcal{S}$.

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. 13: Ideal functionality $\mathcal{F}$