Privacy-Preserving Multi-Party Keyword-Based Classification of Unstructured Text Data

Ian Pépin
School of Computing
Queen's University
Kingston, ON, Canada
ian.pepin@queensu.ca

Furkan Alaca
School of Computing
Queen's University
Kingston, ON, Canada
furkan.alaca@queensu.ca

Farhana Zulkernine
School of Computing
Queen's University
Kingston, ON, Canada
farhana.zulkernine@queensu.ca

***Abstract*—Collaborative data analytics and text classification are required in numerous application domains including health data analytics, which has strict privacy constraints. However, sharing data with other parties widens the attack surface and thus increases the risk of the data being compromised. Research on applications of secure multi-party computation to health data analytics have mostly focused on structured text or numeric data. In this work, we formulate a privacy-preserving multi-party scheme for performing keyword-based classification of unstructured text data, with a case study on medical text data. Our scheme uses arithmetic secret sharing and we implement it using the CrypTen framework. We also devise techniques that significantly reduce computation time without impacting accuracy, enhancing the practical feasibility of our approach.**

***Index Terms*—Protected Health Information, PHI, Secure Multi-Party Computation, MPC, Secret Sharing, Natural Language Processing, NLP, privacy**

I. Introduction

Data breaches are at an all-time high, and the average total cost of a data breach rose to US$4.45 million in 2023. Many breaches are caused by phishing, compromised credentials, or insider threats [1]. Data breaches pose increased risks to parties that share data with others for collaborative analysis. Medical data in particular contains sensitive information about patients, and is shared with researchers and healthcare professionals for analysis. However, shared data can be exposed if one or more parties are compromised or if they breach trust. Collaborating parties may also uncover the identity of people who are not present in their own dataset when linking their data with each other [2]. A trusted third party (TTP) may be relied upon to collect and analyze the data from the collaborating parties and share the output. However, this requires the collaborating parties to trust the TTP to perform the analysis faithfully without breaching trust [3] and to not fall victim to attacks that can expose the private data [4].

Cryptographic protocols can enable collaborative analysis of distributed data while protecting the confidentiality of the collaborating parties' data from each other. Secure Multi-Party Computation (MPC) allows two or more parties to jointly compute a function on their encrypted inputs and obtain the output while keeping their inputs private. For example, one party can compare a person's DNA against a database of cancer patients' DNA held by another party to determine if the person is in a high-risk cancer group [5]; neither party learns any information about the other party's data except for the output, which is only the category of cancer closest to the person's DNA. Two or more parties can also use MPC to perform machine learning tasks on their private data [2] [6].

We propose and implement a collaborative Privacy-Preserving Text Classification scheme for distributed medical text data using Secret Sharing (PPTC-SS). Secret sharing allows collaborating parties to provide each other with transformed versions of their data that do not reveal any meaningful information about the original data. The parties perform the analysis task on the transformed data, and obtain the final output by combining the results of the computations on the transformed data. Secret sharing thus offers security benefits over conventional data sharing, since all parties' original (plaintext) data remain confidential.

We evaluate our scheme on a real-life analysis task involving unstructured medical chart notes of arthritis patients. The analysis task is to find patients affected with knee or hip arthritis, which are the commonly affected bone joints in the arthritis patient population in primary care settings [7]. Knowledge of the affected bone joints can help determine appropriate treatments such as drug prescriptions, physiotherapy, or medical imaging referrals. This analysis task illustrates the utility of patient-classification based on unstructured medical text data, and can be generalized to other use case scenarios involving different domain-specific text data. We present both the conventional data sharing and analysis method and our proposed PPTC-SS method for performing this task. We demonstrate the feasibility and utility of our proposed PPTC-SS method by a comparative evaluation of its performance against the conventional method. We also devise optimizations to improve the computation time and reduce the communication overhead of our proposed PPTC-SS method.

The rest of this paper is organized as follows. Section II introduces necessary background on MPC methods and related work on applications of MPC to medical data analytics. Section III describes the medical data analytics scenario, the design of the conventional data sharing and analytics approach, and the proposed MPC-based PPTC-SS approach. Section IV describes our implementation. Section V presents and discusses our experimental results. Section VI concludes and

Version: August 31, 2024. This is the authors' version of the work. The definitive Version of Record is published in the proceedings of the 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT 2024) and is available on IEEE Xplore: https://ieeexplore.ieee.org/document/10621516 (DOI link: https://doi.org/10.1109/DCOSS-IoT61029.2024.00037).

outlines some future research directions.

II. Background and Related Work

We first introduce some of the primitives used in MPC protocols, followed by related work that applies MPC to medical data analytics. We then discuss challenges and prior work on unstructured medical text classification.

Oblivious Transfer (OT) is a two-party protocol [8] where a sender has m messages and a receiver selects k messages to transfer without the sender knowing which message(s) it sent [9]. Its simplest form is a 1-out-of-2 scheme where the sender has two messages m_0, m_1 and the receiver selects a bit $b \in \{0, 1\}$. OT guarantees that the receiver obtains m_b without learning anything about m_{1-b}, while the sender does not learn anything about the receiver's selected bit b.

OT is used as a building block for other MPC primitives such as garbled circuits. OT is also used in Private Information Retrieval (PIR) to query a database without revealing the queried records to the database server [9] [10].

Garbled Circuits (GC) is a two-party protocol where the parties evaluate an encrypted Boolean circuit to compute an arbitrary function over their encrypted inputs [11]. The evaluation of a garbled circuit does not leak any information to either party about the other party's inputs [12].

Garbled circuits have been applied to calculate the edit distance between a patient and a disease diagnosis on genomics data [13] [14]. They have also been used with deep learning to predict patient diagnoses [15] [16] [17]. Lazrig et al. [4] proposed a privacy-preserving record linkage (PPRL) scheme to identify records that belong to the same individual in two different datasets containing structured medical data. They find matching records by using Bloom filters and garbled circuits to calculate the Dice coefficient of every pair of records across the datasets. Aside from the output, no other information is revealed to either party. It takes approximately eight hours to find matches between two databases of 10,000 records.

Homomorphic Encryption (HE) allows users to perform computations on ciphertext such that decrypting the result yields a value identical to performing the same computations on the original plaintext data. For example, adding two encrypted integers would result in the same outcome (when decrypted) as adding the same integers in unencrypted form.

HE can be used by clients to store encrypted data on an untrusted server [18] [19], which is unable to decrypt the data without the client's private key. The client can then query the server to perform computations on the encrypted data and return the encrypted result. This saves the client from downloading and decrypting the data to compute on it locally.

HE can also be used to train and evaluate machine learning models on encrypted data [20] [21] [22] to preserve the confidentiality of the input data. The model can also be encrypted and shared to allow other parties to apply it to their data without learning the encrypted model's weights.

Secret Sharing allows a secret to be divided into an arbitrary number of *shares* and distributed among multiple parties, such that no individual party learns any meaningful information about the secret. A party (or an attacker) can only reconstruct a secret if they collect a predetermined threshold number of shares from the other parties. Blakley [23] and Shamir [24] first introduced this idea independently in 1979.

Rogers et al. [25] proposed VaultDB, a framework to securely execute SQL queries on two or more structured medical data sources using a two-party protocol. One application is to identify demographic groups that may have untreated or undertreated hypertension in a patient population. Their framework can handle up to one million input tuples efficiently, and the protocol scales well as the amount of data increases. Van Egmond et al. [26] developed a method to combine distributed datasets and perform Lasso regression to identify high-impact lifestyle factors of heart failure in Dutch patients. They use Private Set Intersection (PSI) to identify records that appear in two distributed datasets and use secret sharing to perform their computations.

We use secret sharing in our proposed scheme, since it is more easily scalable to an arbitrary number of parties [6]. This is advantageous for medical applications, since patients may visit multiple hospitals and clinics. In secret sharing, the collaborating parties each compute a function on shares of the encrypted data, and the resulting outputs from all parties need to be combined to decrypt the outcome of the function; this is in contrast to HE, in which one of the parties independently evaluates a function on all the encrypted data.

In this work, we demonstrate our proposed approach using keyword-based classification of unstructured medical text data to classify the type of septic arthritis as knee or hip or both. Very few works exist in the literature that perform keyword-based classification of unstructured medical text data [7] [27] due to the inherent challenges of processing domain specific acronyms and medical vocabulary, and absence of labelled training data required to train machine learning models for classification. There have been more work in automatic keyword extraction to find words that correlate with certain phenomena [28] [27]. Luo et al. [29] and Judd et al. [30] proposed email classification and back pain classification models using topic modelling and keyword based approaches respectively. Topic models are trained on labelled data but can be used to first cluster text corpora for later application of keyword specific categorization. Back pains have specific medical terms and disease codes associated to classify location and pain levels which are used in keyword based classification. Generally, in natural language multiple synonyms can exist for chosen words. In that case all synonyms can be used with word embedding or vectorization techniques to classify text data [7]. Since our work is more focused on achieving our security goals, we use very simple keywords such as 'hip' and 'knee' to search for relevant text in Electronic Medical Records (EMR) and apply word embeddings to capture more contextual information (e.g., 'no pain in knee', 'hip fracture') to classify patients having knee pain, hip pain, both knee and hip pain, or neither.

III. Design Overview

Medical data may be structured or unstructured and appear in different schema across multiple distributed sources, such as EMRs in primary care and hospital databases. Handling this data requires abiding by stringent ethics and privacy policies to protect patient privacy [31]. We consider a setting consisting of three parties, where the first holds unstructured medical data of patients, the second holds the criteria (e.g., keywords) for classifying patients, and the third develops and holds an algorithm to run on *Party-1*'s data with the criteria from *Party-2* as an input to the algorithm. We map this setting to a real-world scenario as follows. *Party-1* is a clinic or hospital that holds unstructured medical notes spanning multiple years of data from patients' visits or encounters with primary care providers. *Party-2* is an insurance company that requires medical information from *Party-1*'s data about individuals who meet certain criteria to approve/reject insurance applications or claims. *Party-3* is a healthcare data analyst who has an algorithm to extract patients' information from *Party-1*'s data given some criteria, which is provided by *Party-2* in our use case scenario. *Party-3* does not contribute its own data to the analysis task.

We assume that *Party-1* and *Party-2* each have Data Custodians (DC), as is common in health research settings [32], who hold the raw data and can de-identify it if applicable, before sending it to a Data Scientist (DS). Medical data is commonly de-identified to protect patient privacy before it is shared [33] [34]. De-identification entails removing Protected Health Information (PHI) such as names, addresses, phone numbers, and birthdates or replacing them with masked data. A unique identifier is often assigned to allow the de-identified data to be linked back to the patient.

Below, we present *Approach-1*, a conventional plaintext computation approach, and *Approach-2*, our proposed MPC-based computation approach using secret sharing. Both of the approaches map to the three-party scenario that we described above, but *Approach-2* offers additional security benefits that *Approach-1* does not. To compare the performance of the two approaches, we select a case study for which we perform our experiments. In our case study, *Party-3* determines the affected bone joints of a specific patient by employing a rule-based classification technique that uses a set of keywords from *Party-2* to match against a selected arthritis patient's unstructured medical notes from *Party-1*. This scenario generalizes to other application domains, where *Party-1* holds unstructured text data and *Party-2* holds keywords that can be matched against *Party-1*'s data by computing pairwise similarity measures between words. This scenario can also be extended to settings where the data is distributed across more parties: instantiating additional parties with the same function as *Party-1* or *Party-2* changes little in the overall data analysis method described in the following sections.

A. *Approach-1: Conventional Plaintext Computation*

The conventional plaintext computation approach is executed as follows and is illustrated in Fig. 1. First, in

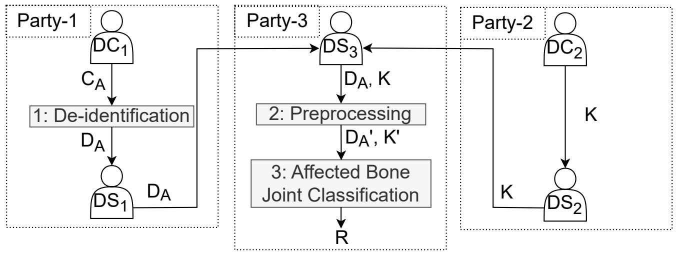

Fig. 1. Execution of *Approach-1*: Conventional plaintext computation. Grey rectangles represent operations performed by an individual within the party that it is located in. Arrows represent information flow and are labelled with the data that is output by an operation or sent from one individual to another.

Step 1, the data custodian of *Party-1*, DC_1, retrieves the data record of all medical notes corresponding to patient A, $C_A = \{c_{A_1}, c_{A_2}, ..., c_{A_N}\}$, where N is the number of words contained in the record. DC_1 then de-identifies the data to produce $D_A = \{d_{A_1}, d_{A_2}, ..., d_{A_N}\}$, and sends it to the data scientist, DS_1. As the data custodian, DC_1 stores data of other patients too (e.g., C_B would denote patient B's data record), but since the analysis task in this paper only operates on one patient's data at a time, we simplify our notation by focusing on a single patient A when presenting our scheme. The data custodian from *Party-2*, DC_2, retrieves the lists of keywords $K = \{K_1, K_2, \dots, K_L\}$, where each $K_i \in K$ is a set of keywords corresponding to a diagnosis label (e.g., knee, hip) and L is the number of labels. K does not contain PHI in our setting, and thus DC_2 sends it to the data scientist DS_2 without de-identification. DS_1 and DS_2 then send D_A and K, respectively, to the data scientist DS_3 of *Party-3*. DS_3 preprocesses them in Step 2 to produce ${D_A}' = \{{d_{A_1}}', {d_{A_2}}', ..., {d_{A_N}}'\}$ and $K' = \{{K_1}', {K_2}', \dots, {K_L}'\}$ respectively, and uses the data to determine the affected bone joints for the specified patient A in Step 3. A one-hot result vector R is produced, each bit of which corresponds to a class label. A predicted class is indicated by the bit value of 1 with the remaining bit values being 0. Since each patient can have any number of diagnoses (e.g., in our setting, either knee or hip arthritis, or both), or no diagnosis at all, the number of elements in R equals the cardinality of the power set of K, i.e., $|\mathcal{P}(K)| = 2^{|K|}$. Thus, in our affected bone joint classification task, our result vector takes the form $R = (r_{\text{knee}}, r_{\text{hip}}, r_{\text{kneeAndHip}}, r_{\text{other}})$.

1) Keyword matching methods: Party 3 preprocesses the de-identified patient record D_A and the keywords K by applying stop word removal, tokenization, lemmatization, and chunking, and converts the resulting tokens into either word embeddings or hashes (discussed below) to produce ${D_A}'$ and K'. Then, the affected bone joint classification task identifies whether any word tokens in ${D_A}'$ match any keyword tokens from K'. If the patient's notes contain at least one knee-related token and at least one hip-related token, we set $r_{kneeAndHip} = 1$. Otherwise, we set either $r_{knee} = 1$ or $r_{hip} = 1$ if the patient's notes contain at least one hip-related or at least one knee-related token, respectively. If no keyword matches are found, we set $r_{other} = 1$. We evaluate two methods of keyword matching as described below, which we use when ${D_A}'$ and K' contain either word embeddings or hashes, respectively.

Word embeddings: Embeddings encapsulate semantic similarities among words in a vector space. *Party-3* determines the similarity between each word token embedding $d_{A_i} \in {D_A}'$ and each keyword token embedding $k_j \in K'$ by calculating the cosine similarity, defined as

$$S_C({d_{A_i}}', k_j) = \frac{{d_{A_i}}' \cdot k_j}{\|{d_{A_i}}'\| \cdot \|k_j\|},$$

which yields a value between 0 and 1; values closer to 1 indicate higher similarity. We then apply a threshold to determine if each keyword token in K' is present in ${D_A}'$. We do this by setting the threshold value to 0.95 to ensure that only exact keyword matches meet the threshold, for the purpose of allowing a like-to-like comparison of computation time with our hashing-based scheme as described below. In practice, the threshold should be tuned to improve classification accuracy.

Hashing: Using hashes instead of word embeddings reduces data size, which helps to reduce computation and communication overhead with MPC. We use the BLAKE2 [35] hash function to compute the hashes in ${D_A}'$ and K'. However, word similarity cannot be determined when comparing hashes; we can only determine word equality. To find matching keywords, *Party-3* subtracts the hashes in ${D_A}'$ and K' and compares the result to zero to determine if a match has been found.

2) Security and Privacy Limitations of Approach-1: In this approach, *Party-3* has access to *Party-1*'s clinical notes, which although de-identified, contain information about patients, their healthcare providers, other hospital or clinic staff, and even family members who may not have consented to have their identity used in research. Although there is an agreement between the parties that allows *Party-3* to use the clinical notes, the patients' identity, their family history, and their healthcare providers' identity are still exposed to another (potentially corrupted) party [36]. *Party-1* has to trust that *Party-3* will not abuse its power and correctly handle the clinical notes without disclosing any PHI to unauthorized parties [3]. Moreover, *Party-2* is required to disclose its classification criteria, which it may prefer to keep confidential for business or operational reasons. All the data sent by *Party-1* and *Party-2* (${D_A}'$ and K', respectively) are stored by *Party-3*, and thus an attacker that compromises or colludes with *Party-3* will have access to all the data.

B. Approach-2: Secure Multi-Party Computation

We now discuss the security goals, threat model, and the data flow (illustrated in Fig. 2) of our proposed MPC-based PPTC-SS approach.

1) Security Goals: We employ secret sharing to protect the confidentiality of *Party-1*'s and *Party-2*'s data from each other. Thus, *Party-1* and *Party-2* should learn only the size of each other's inputs but nothing else. *Party-3* should only learn the output of the affected bone joint classification task; it should not learn the plaintext input data from *Party-1* or *Party-2*.

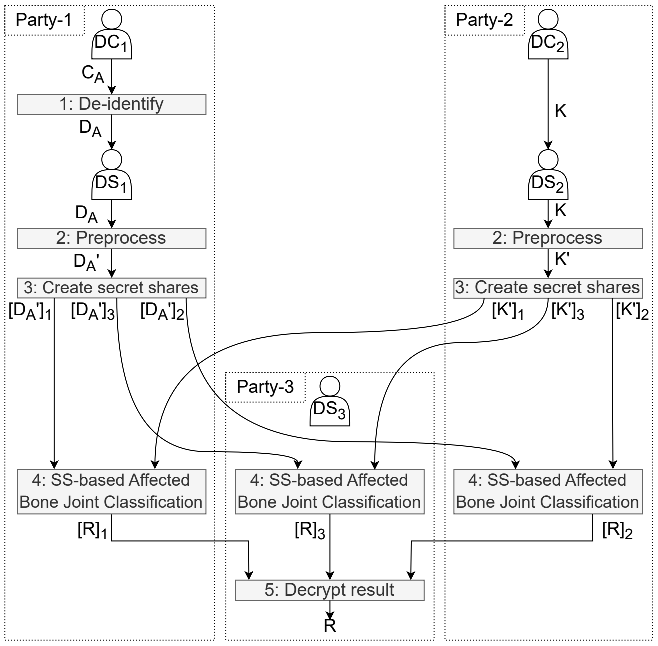

Fig. 2. Execution of *Approach-2*: MPC with secret sharing. Grey rectangles represent operations performed by an individual within the party that it is located in. Data custodians (DC) obtain the initial data and de-identify it if applicable; data scientists (DS) perform all subsequent operations. Arrows represent information flow and are labelled with the data that is output by an operation or sent from one individual to another.

2) Threat Model: We assume that communication channels between the parties are secured, e.g., using Transport Layer Security (TLS). We assume a semi-honest adversary model where the parties do not deviate from the protocol. In medical and other applications where malicious behaviour can lead to serious legal, reputational, and financial consequences, a semi-honest adversary model is sufficient since the entities involved have little incentive to risk their reputation by behaving maliciously or by colluding with each other [37] [38] [39] [40] [41]. We also aim to ensure that an adversary that corrupts one party to access its internal state and the data it receives during the computation should not be capable of stealing or inferring information about the other honest parties' data (e.g., they should not be capable of inferring the patients' diagnoses)—this ensures that a data breach of one party does not lead to the compromise of other parties' data or to any data leakage from the execution of the protocol [5].

3) PPTC-SS Data Flow: As shown in Fig. 2, after Step 1, DS_1 has patient A's de-identified record D_A and DS_2 has the keywords K, as was the case in *Approach-1*. Next, in Step 2, *Party-1* preprocesses D_A to produce ${D_A}'$ and *Party-2* preprocesses K to produce K'. In Step 3, *Party-1* splits ${D_A}'$ into three secret shares $[{D_A}']_1$, $[{D_A}']_2$, and $[{D_A}']_3$. The square bracket notation indicates that the element is a secret share. *Party-1* keeps $[{D_A}']_1$ for itself and sends the other two shares to *Party-2* and *Party-3*, respectively. Similarly, *Party-2* splits K' into three shares $[K']_1$, $[K']_2$, and $[K']_3$, keeps $[K']_2$ for itself, and sends the other two shares to *Party-1* and *Party-3*, respectively.

In Step 4, three-party computation is performed to execute the affected bone joint classification, with each party executing the operations described in Section III-A on their respective secret shares. Some of the operations (e.g., computing cosine similarities between the secret shared values) require communication rounds between parties, which take place during Step 4 of Fig. 2. Each party produces its corresponding secret share of the result vector $[R]_p = \{[r_{\text{knee}}]_p, [r_{\text{hip}}]_p, [r_{\text{kneeAndHip}}]_p, [r_{\text{other}}]_p\}$, where $p \in \{1, 2, 3\}$. *Party-1* and *Party-2* then send $[R]_1$ and $[R]_2$ respectively to *Party-3*, which combines them with $[R]_3$ to decrypt the final result vector $R = \{r_{\text{knee}}, r_{\text{hip}}, r_{\text{kneeAndHip}}, r_{\text{other}}\}$, as depicted in Step 5 of Fig. 2. Thus, only *Party-3* learns whether any matches were found between *Party-1*'s clinical notes and *Party-2*'s keywords, which in our scenario reveals whether the patient has septic arthritis in the knee, hip, both, or in another joint.

4) Protection of Data at Rest: The extraction and de-identification of patients' data (Step 1), preprocessing (Step 2), and the creation and distribution of secret shares (Step 3) can all be done in advance. After the distribution of secret shares, DS_1 may delete the plaintext medical notes of all patients and keep only its secret share of the patients' medical notes, and similarly DS_2 may delete its plaintext keywords and keep only its secret share of the keywords. This offers additional protection for data at rest, since an adversary would need to break into all three of DS_1, DS_2, and DS_3 to obtain all of their secret shares to reconstruct the medical notes and the keywords. However, compromising DC_1 or DC_2 would result in the exposure of the medical notes or keywords, respectively.

Since *Party-3* does not provide its own input data to the affected bone joint classification, it is possible for *Party-1* and *Party-2* to perform two-party computation and send their resulting secret shares to *Party-3*. This would save communication overhead compared to three-party computation, and thus speed up the execution time. However, three-party computation has a security advantage for protecting data at rest, since an attacker would need to compromise three parties instead of two parties to obtain all the secret shares.

C. Optimizations

We propose two optimizations to reduce the execution time of *Approach-2*, as follows.

Optimization 1: Remove conditionals. When evaluating conditional statements on secret-shared data, unlike conventional if-statements, both results of the condition must be evaluated to avoid leaking data [6]. To avoid this overhead, this optimization removes conditional statements on secret-shared values by performing MPC only for the similarity score calculations, followed by *Party-3* performing the thresholding step. Thus, all parties compute the similarity scores for each secret-shared word token in ${D_A}'$ against each secret-shared keyword in K'. *Party-1* and *Party-2* send their resulting secret shares to *Party-3*. *Party-3* then decrypts the similarity scores by combining the secret shares, and performs the word matching by applying the similarity threshold. If the hashing method is used for word matching, the procedure for this optimization is similar: *Party-1* and *Party-2* use their secret shares to compute the difference between each secret-shared word token in ${D_A}'$ and each secret-shared keyword in K'. Then, *Party-3* collects all the shares to decrypt the results and compares each value to 0 to determine matched keywords. While this optimization removes the comparatively more expensive conditional operations, it introduces additional decryption operations.

The disadvantage of this optimization from a security perspective is that *Party-3* learns all the similarity scores. Thus, limiting *Party-3*'s ability to reconstruct the medical notes requires mitigations. We leave this to future work to explore, e.g., by using techniques such as truncating the similarity scores to the fewest number of significant digits required for the thresholding step or adding random perturbations [42] to the similarity scores.

Optimization 2: Stop decryption early. To reduce decryption time, we reduce the number of decryption operations as follows. We split the similarity scores into L batches. Each batch represents the distances between all of *Party-2*'s keywords corresponding to a single diagnosis label (i.e., all keywords in $K_i \in K'$ from *Party-2*, where $1 \leq i \leq L$) and all the word tokens in patient A's clinical notes in ${D_A}'$ from *Party-1*. If *Party-3* finds a keyword match in one batch, it concludes that the patient has the corresponding diagnosis and skips the decryption of the secret shares remaining in that batch. *Party-3* then proceeds to decrypting the batch corresponding to the keywords for the next diagnosis label, ${K_{i+1}}'$. If the hashing method is used for word matching, the procedure for this optimization is the same.

Optimization 2 thus reduces execution time for patients with a positive diagnosis, since finding a keyword match means that the remaining secret shares in the batch do not need to be decrypted. Thus, measuring the execution time or power consumption of *Party-3* may help an adversary infer information about the patient's diagnosis. For example, *Party-1* and *Party-2* may infer that the patient has a positive diagnosis if they determine that *Party-3* computed the results more quickly. Countermeasures are thus required to prevent information leakage, e.g., by delaying *Party-3* from signalling to *Party-1* and *Party-2* that the analysis task is complete.

IV. Implementation

We use PyTorch to implement our conventional scheme and the CrypTen [6] framework to implement our proposed MPC-based PPTC-SS scheme. CrypTen supports arithmetic and Boolean secret sharing with an arbitrary number of parties to perform computations on their private datasets. All secret sharing operations occur in the ring $\mathbb{Z}_{2^{64}}$. CrypTen is secure against semi-honest adversaries that can corrupt up to $n-1$ parties, satisfying our threat model stated in Section III-B2.

All binary and arithmetic secret sharing primitives implemented in CrypTen are proven to be secure, and functions that are compositions of these secure primitives are also secure

[6]. CrypTen uses private addition and multiplication on secret shares to implement all other operations; non-linear functions such as cosine (which we use to compute cosine similarity) are implemented using numerical approximations that rely on only addition and multiplication [6]. For efficiency, CrypTen relies on a Trusted Third Party (TTP) to provide multiplication triples during the computations, but the TTP otherwise does not engage in the protocol. The TTP is assumed to be semi-honest and able to correctly generate the triples. A TTP-free solution relying on either homomorphic encryption or oblivious transfer is planned [6].

A. Dataset Extraction and Preprocessing

We extract the dataset for *Party-1*'s unstructured text data from the MIMIC-III database [43], which contains de-identified patient data that is similar to primary care data [7]. We select all 90 patients from the database who are diagnosed with knee septic arthritis and hip septic arthritis [44], of which 29 have clinical notes and 87 have medication data. Of these patients, 30 have hip septic arthritis, 64 have knee septic arthritis, and 4 have both diagnoses. We select the "Nursing" category in the "NOTEEVENTS" table of the MIMIC-III database to extract the unstructured data. Fig. 3 shows a sample synthetic clinical note of a patient after it is de-identified.

> Patient is due for knee surgery and knee replacement on [**2012-09-07**], was referred to Dr [**Last Name (STitle) 1**] [**Name (STitle) 2**] by Dr [**Last Name (STitle) 3**], he prescribed Oxycontin 10 mg, she uses 1 a day. Wearing braces on both knees. [**09-28**] lungs clear, severe OA knees. BP 120/80, HR 89.

Fig. 3. A sample de-identified clinical note.

Party-2 has a set of keywords that indicate the affected bone joints of the arthritis patient, which will be matched against *Party-1*'s unstructured text data. We use two sets of 12 keywords each for knee septic arthritis and hip septic arthritis, respectively. The keywords consist of 1-word or 2-word strings such as "hip pain", "kneeseptic", and "arthritis knee".

We preprocess all the text data as follows. We apply stop word removal, tokenization, lemmatization, and chunking to the text data. Finally, depending on the configuration being evaluated, we either use BioWordVec [45] to convert the word tokens into 200-dimensional vector embeddings, or we use the BLAKE2 hash function to hash the words.

We truncated the BLAKE2 hashes to 32 bits, which offers sufficient collision-resistance for our dataset and maximizes performance since it can be stored in a single integer in a PyTorch tensor; longer hashes can be used if required, at the expense of speed. Approximately 77,162 unique words are needed for a collision to occur with a probability of 0.5, as computed by $2^{(n+1)/2}\sqrt{\ln\frac{1}{1-\lambda}}$, where n is the bit-length of the hashes (32) and λ is the collision probability (0.5) [46]. There are 208,576 words in total (6,251 of which are unique) in the clinical notes from the sample of patients we evaluated on, and no hash collisions occurred in our experiments.

TABLE I
CONFUSION MATRIX FOR THE AFFECTED BONE JOINT CLASSIFICATION.

		Actual			
		Knee	**Hip**	**Both**	**Other**
Predicted	**Knee**	1	0	0	0
	Hip	1	10	0	0
	Both	0	0	1	0
	Other	6	10	0	0

B. Experimental Setup

We deployed our implementation on a single machine with an Intel i5-1035G1 processor and 8GB of RAM, with *Party-1* and *Party-2* each running in a virtual machine (VM) and *Party-3* running on the host machine. The host machine and both VMs ran Ubuntu 22.04, Python 3.11, and CrypTen 0.4.1. The parties communicated over bridged virtual network interfaces. Since CrypTen is built on PyTorch, we used PyTorch functions for communication between the parties.

V. VALIDATION AND RESULTS

We evaluate our proposed PPTC-SS approach and the conventional baseline approach based on the execution time required to determine an individual patient diagnoses of both hip and knee septic arthritis. We also evaluate the impact of our two proposed performance optimizations on execution time. We execute each experiment five times, randomizing the order of *Party-2*'s keywords in each iteration, and report the average total execution time and communication statistics (measured using CrypTen) in Table II.

A. Classification Performance Metrics

The weighted average of the accuracy, recall, precision and F1-Score metrics for the affected bone joint classification are 0.67, 0.41, 0.94, and 0.54, respectively. We obtain these values by computing the metrics for each class (hip, knee, hip and knee, and other) and then computing their weighted average based on the number of true occurrences for each class. We present the confusion matrix in Table I. Both the PyTorch (plaintext) and the CrypTen (MPC) configurations yield the same results, confirming that the secret sharing operations do not change the classification results. The high precision indicates that false positives are low. Accuracy may be improved, e.g., by better data cleaning and preprocessing, tuning the similarity threshold for keyword matching, or tuning the keywords required to determine a diagnosis. However, our focus in this study is on the design and performance of the proposed MPC-based classification approach.

B. Performance of Approach-1 and Approach-2

The performances of our conventional *Approach-1* and the PPTC-SS *Approach-2* using MPC for affected bone joint classification are shown in Table II, with and without the two optimizations. For each configuration, we show the number of keywords from *Party-2* that are used for the comparisons, the total number of comparisons between the words from the clinical notes and the keywords, the comparison time, and

TABLE II

EXECUTION TIME AND COMMUNICATION STATISTICS OF THE AFFECTED BONE JOINT CLASSIFICATION TASK USING PYTORCH AND CRYPTEN ON A SINGLE PATIENT NOTE CONTAINING 5,741 WORDS. ALL TIMES ARE MEASURED AS WALL-CLOCK TIME. THE BYTES COMM. AND COMM. TIME COLUMNS REFER TO THE BYTES COMMUNICATED AND TIME SPENT COMMUNICATING, RESPECTIVELY, AVERAGED ACROSS THE THREE PARTIES.

Configuration	**Keywords Compared**	**Number of Comparisons**	**Comparison Time (s)**	**Comparisons Per Second**	**Comm. Rounds**	**Bytes Comm.**	**Comm. Time (s)**	**Total Time (s)**
1. PyTorch: Embeddings	24	137,784	4.53	30,415.9	3	27	0.0066	**10.57**
2. PyTorch: Hash Values	24	137,784	0.86	160,214.0	3	11	0.0048	**16.84**
3. CrypTen: Embeddings	24	137,784	177.45	776.5	13,473	6,048,850,912	132.56	**197.25**
4. CrypTen: Embeddings & Opt. 1	24	137,784	133.01	1,035.9	8,576	5,678,892,939	95.48	**152.40**
5. CrypTen: Embeddings & Opt. 2	12	68,892	95.19	723.7	9,810	3,292,532,262	73.44	**112.35**
6. CrypTen: Embeddings & Opt. 1 and 2	12	68,892	71.85	958.8	7,301	3,092,077,860	55.43	**89.45**
7. CrypTen: Hash Values	24	137,784	41.60	3,312.1	10,341	372,819,488	38.95	**58.30**
8. CrypTen: Hash Values & Opt. 1	24	137,784	3.04	45,323.7	5,804	2,116,619	11.46	**28.26**
9. CrypTen: Hash Values & Opt. 2	7.6	43,632	13.93	3,132.2	7,173	118,501,198	17.78	**31.31**
10. CrypTen: Hash Values & Opt. 1 and 2	6.8	39,039	0.80	48,798.8	5,787	1,063,420	7.18	**17.22**

the number of comparisons per second. The total (wall-clock) time is the sum of computation time, preprocessing time, and communication time between parties.

The conventional *Approach-1* configurations 1 and 2 complete in 10.57s using word embeddings and 16.84s using hashes, respectively. These conventional configurations were 18x and 3x faster than the MPC-based *Approach-2* configurations respectively: Configuration 3 completed in 197.25s using word embeddings and Configuration 7 completed in 58.30s using hashes. The MPC-based configurations incur significant communication overhead, which increases the total runtime. For example, Configuration 3 incurs 13,473 communication rounds and approximately 6 GB of communication per party, compared to only 3 rounds and 27 bytes for Configuration 1. The embeddings-based configurations incur higher communication cost compared to hash-based configurations; this is explained by the higher dimensionality of the embeddings and the higher cost of computing cosine similarity. Medical notes represented with embeddings are stored in $N \times 200$ tensors whereas medical notes represented with hashes are stored as $N \times 1$ tensors, where N is the number of words.

Our optimizations reduce the total wall-clock time of our proposed *Approach-2* from 197.25s (Configuration 3) to 89.45s (Configuration 6) when using word embeddings, and from 58.30s (Configuration 7) to 17.22s (Configuration 10) when using hashes.

Optimization 1 increases the comparisons per second from 3,312.1 to 45,323.7 between configurations 7 and 8, which use hashes. However, it yields a smaller improvement with embeddings, from 776.5 to 1,035.9 comparisons per second between configurations 3 and 4. This is because CrypTen's numerical approximation of cosine similarity is slower than the subtraction operation of two hash values. These results confirm the net performance improvement achieved by the removal of the conditional statements on secret-shared values and the addition of less costly decryption operations. This improvement can also be observed in the communication statistics, with a proportionally greater improvement for the hash-based configuration than for the embedding-based configuration.

Optimization 2 reduces the number of total comparisons and the total wall-clock time of *Approach-1* and *Approach-2* if the patient has a positive diagnosis of either knee or hip septic arthritis. For a patient with both knee and hip septic arthritis, the number of comparisons is reduced from 137,784 to 68,892 between configurations 3 and 5, which use embeddings, and from 137,784 to 43,632 between configurations 7 and 9, which use hashes. However, Optimization 2 does not speed up negative diagnoses, since all the comparisons must be performed to reach a negative diagnosis.

VI. CONCLUDING REMARKS

This study implements an MPC-based approach for privacy-preserving classification of unstructured text data using classification criteria stored by another party. We evaluate our scheme on a representative medical scenario, in which a data analysis service provider (*Party-3*) can provide a service to classify a patient based on their private medical data stored by a primary care clinic or a hospital (*Party-1*) using confidential classification criteria of an insurance provider (*Party-2*). Our scheme uses secret sharing to enable the execution of the classification task while maintaining the confidentiality of all parties' data. While our proposed approach is more computationally expensive than conventional computation on plaintext data, it ensures that no party has access to other parties' plaintext data. Our proposed approach offers additional protection in case of a data breach of any single party. Our scheme generalizes to other scenarios that require classification of unstructured text data based on the criteria specified by another party, and which share the same confidentiality requirements as the scenario we presented.

While our current study sheds light on the feasibility of privacy-preserving text classification, there remain avenues for future exploration. While we employed a more simple rule-based classification technique, the feasibility of employing other NLP-based text classification techniques can be explored [7] [27]. Additional performance optimizations can also be explored. Trade-offs between classification accuracy and computational performance may differ for different classification techniques when executing them with MPC, compared to plaintext computation, due to the different cost characteristics of executing various operations such as conditional statements under MPC. Future work can thus investigate which classification techniques offer the best overall performance and accuracy characteristics for MPC.

References

[1] IBM Security. (2023) Cost of a data breach 2023. Accessed on Sep. 30, 2023. [Online]. Available: https://www.ibm.com/reports/data-breach

[2] W. Zheng, R. Deng, W. Chen, R. A. Popa, A. Panda, and I. Stoica, "Cerebro: a platform for multi-party cryptographic collaborative learning," in *USENIX Security Symp.*, 2021, pp. 2723–2740.

[3] T. Locher, S. Obermeier, and Y. A. Pignolet, "When can a distributed ledger replace a trusted third party?" in *IEEE Int. Conf. Internet of Things and IEEE Green Comput. and Commun. and IEEE Cyber, Physical and Social Comput. and IEEE Smart Data*, 2018.

[4] I. Lazrig, T. C. Ong, I. Ray, I. Ray, X. Jiang, and J. Vaidya, "Privacy preserving probabilistic record linkage without trusted third party," in *IEEE Conf. Privacy, Security and Trust*, 2018.

[5] Y. Lindell, "Secure multiparty computation," *Commun. ACM*, vol. 64, no. 1, pp. 86–96, 2020.

[6] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," *Advances in Neural Inf. Process. Syst.*, vol. 34, 2021.

[7] Y. Chen, B. Shu, M. Moattari, F. Zulkernine, J. Queenan, and D. Barber, "SPaDe: A synonym-based pain-level detection tool for osteoarthritis," in *IEEE Int. Conf. Digital Health*, 2023.

[8] M. O. Rabin, "How to exchange secrets with oblivious transfer," *Cryptology ePrint Archive*, 2005.

[9] V. K. Yadav, N. Andola, S. Verma, and S. Venkatesan, "A survey of oblivious transfer protocol," *ACM Comput. Surv.*, vol. 54, no. 10s, pp. 1–37, 2022.

[10] S. Yekhanin, "Private information retrieval," *Commun. ACM*, vol. 53, no. 4, pp. 68–73, 2010.

[11] A. C.-C. Yao, "How to generate and exchange secrets," in *IEEE Symp. Found. Comput. Sci.*, 1986.

[12] D. Evans, V. Kolesnikov, M. Rosulek *et al.*, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2-3, pp. 70–246, 2018.

[13] M. M. A. Aziz, D. Alhadidi, and N. Mohammed, "Secure approximation of edit distance on genomic data," *BMC Med. Genomics*, vol. 10, pp. 55–67, 2017.

[14] M. S. R. Mahdi, M. M. Al Aziz, D. Alhadidi, and N. Mohammed, "Secure similar patients query on encrypted genomic data," *IEEE J. Biomed. and Health Inform.*, vol. 23, no. 6, pp. 2611–2618, 2018.

[15] X. Liu, Y. Zheng, X. Yuan, and X. Yi, "Towards secure and lightweight deep learning as a medical diagnostic service," in *Eur. Symp. Res. in Comput. Security*. Springer, 2021.

[16] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference system for neural networks," in *Workshop on Privacy-Preserving Machine Learning in Practice*, 2020.

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

[18] M. Ibtihal, N. Hassan *et al.*, "Homomorphic encryption as a service for outsourced images in mobile cloud computing environment," in *Cryptography: Breakthroughs in Research and Practice*. IGI Global, 2020, pp. 316–330.

[19] M. Kim, H. T. Lee, S. Ling, and H. Wang, "On the efficiency of fhe-based private queries," *IEEE Trans. Dependable and Secure Computing*, vol. 15, no. 2, pp. 357–363, 2016.

[20] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," *arXiv preprint arXiv:2104.03152*, 2021.

[21] Y. Chen, G. Huang, J. Shi, X. Xie, and Y. Yan, "Rosetta: A privacy-preserving framework based on tensorflow," 2020.

[22] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, vol. 10, pp. 30 039–30 054, 2022.

[23] G. R. Blakley, "Safeguarding cryptographic keys," in *Int. Workshop Managing Requirements Knowledge*, 1979.

[24] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[25] J. Rogers, E. Adetoro, J. Bater, T. Canter, D. Fu, A. Hamilton, A. Hassan, A. Martinez, E. Michalski, V. Mitrovic *et al.*, "VaultDB: A real-world pilot of secure multi-party computation within a clinical research network," *arXiv preprint arXiv:2203.00146*, 2022.

[26] M. B. van Egmond, G. Spini, O. van der Galien, A. IJpma, T. Veugen, W. Kraaij, A. Sangers, T. Rooijakkers, P. Langenkamp, B. Kamphorst *et al.*, "Privacy-preserving dataset combination and lasso regression for healthcare predictions," *BMC Med. Inform. and Decis. Making*, vol. 21, no. 1, pp. 1–16, 2021.

[27] T. Tran, H. Huynh, P. Tran, and D. Truong, "Text classification based on keywords with different thresholds," in *Int. Conf. Intell. Inf. Technol.*, 2019.

[28] A. Reunamo, L. Peltonen, R. Mustonen, M. Saari, T. Salakoski, S. Salanterä, and M. H., "Text classification model explainability for keyword extraction - towards keyword-based summarization of nursing care episodes," *Stud Health Technol Inform*, Jun 2022.

[29] Z. Luo and F. Zulkernine, "An intelligent email classification system," in *IEEE Symp. Ser. Comput. Intell.*, 2023.

[30] M. Judd, F. Zulkernine, B. Wolfram, A. Rajaram, and D. Barber, "Detecting low back pain using text processing and machine learning approaches," in *Int. Workshop on Biological Knowl. Discovery from Data at the Int. Conf. on Database and Expert Syst. Appl.*, 2018.

[31] H. Zafari, L. Kosowan, J. T. Lam, W. Peeler, M. Gasmallah, F. Zulkernine, and A. Singer, "Using deep learning with canadian primary care data for disease diagnosis," *Deep Learning for Biomed. Data Analysis: Techniques, Approaches, and Applications*, pp. 273–310, 2021.

[32] J. Allen, C. Adams, and F. Flack, "The role of data custodians in establishing and maintaining social licence for health research," *Bioethics*, vol. 33, no. 4, p. 502–510, jan 2019.

[33] U.S. Department of Health & Human Services. (2023) De-identification. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html

[34] C. A. Kushida, D. A. Nichols, R. Jadrnicek, R. Miller, J. K. Walsh, and K. Griffin, "Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies," *Med. Care*, vol. 50, no. Suppl, p. S82, 2012.

[35] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2: simpler, smaller, fast as MD5," in *Appl. Cryptography and Netw. Security*, 2013.

[36] M. Abdalla, M. Abdalla, F. Rudzicz, and G. Hirst, "Using word embeddings to improve the privacy of clinical notes," *J. Amer. Med. Inform. Assoc.*, vol. 27, no. 6, pp. 901–907, 2020.

[37] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang, "Outsourceable two-party privacy-preserving biometric authentication," in *ACM Symp. Inf. Comput. and Commun. Security*, 2014.

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

[39] X. Liu, Y. Zheng, X. Yi, and S. Nepal, "Privacy-preserving collaborative analytics on medical time series data," *IEEE Trans. Dependable and Secure Computing*, vol. 19, no. 3, pp. 1687–1702, 2020.

[40] Y. Zheng, H. Cui, C. Wang, and J. Zhou, "Privacy-preserving image denoising from external cloud databases," *IEEE Trans. Inf. Forensics and Security*, vol. 12, no. 6, pp. 1285–1298, 2017.

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

[42] C. Adams, *Introduction to Privacy Enhancing Technologies: A Classification-Based Approach to Understanding PETs*. Springer Nature, 2021.

[43] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[44] M. García-Arias, A. Balsa, and E. M. Mola, "Septic arthritis," *Best Practice & Res. Clin. Rheumatology*, vol. 25, no. 3, pp. 407–421, 2011.

[45] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu, "Biowordvec, improving biomedical word embeddings with subword information and mesh," *Scientific data*, vol. 6, no. 1, p. 52, 2019.

[46] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.