Privacy Preserving Fuzzy Patient Matching Using Homomorphic Encryption

Shiva Ashish Thumparthy
Brainlab
Munich, Germany
email: ashish.thumparthy@brainlab.com

Ilya Sher
Beame.io
Giv'at Shmuel, Israel
email: is@beame.io

Abstract— **Patient record linkage is an important operation that is necessary for identifying similar patients across medical facilities, with the ambition to improve patient outcomes. However, with increased data privacy concerns, these record linkage algorithms must protect sensitive patient demographic information. Previous works have included usage of Bloom filters (susceptible to frequency attacks) and homomorphic encryption (high computational complexity and communication overheads). We propose a record linkage algorithm that utilizes fully homomorphic encryption ciphertext packing for matrices. This ensures that the algorithm remains privacy-preserving and resilient to multiple attack vectors, while allowing multiple patients' records to be compared at once, as opposed to pairwise comparison.**

Keywords- record linkage; homomorphic encryption; ciphertext packing; Bloom filter.

I. Introduction

Medical record interoperability has been a focus of healthcare systems for purposes of providing better patient outcomes. The objective is to take into consideration patients' comprehensive longitudinal medical histories after aggregating their respective records across medical facilities. However, in order to facilitate the exchange of Electronic Medical Records (EMRs), it is essential to know which records correspond to which real-world patient entities, and at which facilities these records reside. In an age of increasing privacy and confidentiality concerns, the comparison of unencrypted patient demographics is obsolete due to strict guidelines mandating protection of sensitive patient information. Therefore, the aim is to perform statistical patient record linkage in a privacy-preserving setting, while maintaining high linkage quality.

The rest of the work is structured as follows. In Section 2, we describe the background of the patient matching problem. Section 3 describes previous solutions and the basis of our work. Section 4 provides the details of our solution. The dataset used and preliminary results obtained are summarized in Sections 5 and 6, respectively.

II. Problem

In jurisdictions without a nationally issued unique identity number, organizations rely on comparisons of patient demographic attributes to establish linkage using quasi-identifiers such as name, date of birth and recent addresses. Comparison of unencrypted demographics, however, leaks sensitive patient information between medical facilities. In addition, this process leaks critical business information to rival healthcare facilities, which is often thought of as a loss of revenue opportunity. Finally, as is often the case while dealing with data from multiple sources, demographics are subject to data entry errors such as character transpositions. Therefore, an important step in realizing this goal is identifying patients with records across facilities through record linkage algorithms, preferably in a statistical, privacy-preserving fashion that is resilient to data entry inconsistencies, i.e., in a probabilistic manner.

III. Solution

Multiple methods to perform patient matching have been proposed. A common building block to encode patient attributes is the Bloom filter. Bloom filters represent a digest of information without containing any original information in the clear. Originally, Bloom filters contained a digest of each individual attribute, i.e., one attribute per Bloom filter. Subsequently, improvements allowed for the representation of an entire demographic record in one Bloom filter, known as a Record Bloom Filter (RBF). However, Bloom filters were subject to scrutiny due to brute force and frequency attacks, especially in the single-attribute case [1].

Homomorphic encryption of Bloom filters has been proposed in literature to overcome this shortcoming. Homomorphic encryption is a form of encryption that allows computation on encrypted data. The result of the computation, when decrypted, is the same as if the operations had been performed on the unencrypted data. The computing party cannot decipher any data it is performing computation on. Homomorphic encryption has been widely thought of as an important block in solving the patient matching problem, as it is a form of encryption that is considered to be resilient to brute force attacks, even in the quantum-computing realm. However, this form of encryption is highly computationally intensive and is accompanied by large communication overheads.

Randall et al. [2] proposed homomorphic encryption of Bloom filters, which would insure against these attacks at the cost of high computational and communication overheads. This work, in addition to Cruz et al. [3], serves as a basis for our solution to patient matching using homomorphic encryption. Finally, we leverage homomorphic encryption ciphertext packing techniques for matrix multiplication, as proposed by Duong et al. [4] to optimize batch comparison.

IV. METHODOLOGY

We have worked on optimizing homomorphic encryption to the patient matching use case. Our primary contribution is a method of matching lists of Bloom filters at once as opposed to pairwise comparisons of encrypted Bloom filters between two facilities. We achieve this using homomorphic encryption ciphertext packing techniques, which allow for encrypting large volumes of data into a single ciphertext. This allows for computations on entire matrices of data as opposed to individual Bloom filter vectors, thereby reducing the number of ciphertexts required from *n* (the number of Bloom filter vectors) to 8.

Encrypted records are compared to arrive at an encrypted binary result (match vs non-match) based on an agreed similarity threshold. This ensures that no parties have access to patient demographics, either in plaintext or through brute force attacks on traditional hash-based encoding schemes.

The protocol makes use of two additional third parties called the Decryption Unit (DU) and the Linkage Unit (LU) [2]. The steps to determine record similarity using the Tversky similarity index for two medical facilities is as follows:

1. The DU generates a homomorphic key-pair, and shares the public key with the facilities and the LU.
2. Each facility breaks patient demographics into bigrams, which are then hashed into one RBF per patient. These RBFs are stacked on top of each other to generate a RBF matrix. The RBF matrix is multiplied with the sum of the numerator and denominator of the Tversky index threshold. Two ciphertexts are then generated from the RBF matrix: the first by encryption, the second by inversion and encryption.
3. Two additional vectors are generated at each facility, with the plaintext count of 1s from each row and each column of the RBF matrix. The count vectors are multiplied with the numerator of the Tversky index threshold. They are then encrypted into one ciphertext each using vector packing techniques. All 4 ciphertexts are sent to the LU.
4. The LU randomly picks the row count and encrypted RBF from one facility. It then picks the column count and the inverted-and-encrypted RBF of the other facility. The two RBF ciphertexts are multiplied to generate the multiplied matrix ciphertext. This is sent to the DU, along with the chosen row and column count ciphertexts.
5. The DU decrypts the received ciphertexts. The row count vector is converted into a row matrix (with same dimensions as the multiplied matrix) by making each element a row in the row matrix. Similarly, the column count vector is converted into a column matrix by making each element a column in the column matrix.
6. The DU then subtracts the generated row and column count matrices from the multiplied matrix to generate the plaintext result matrix, which it forwards to both facilities.

V. DATA SOURCES

The patient matching problem in the privacy-preserving setting is hard to realize and implement in the real world. This is mainly due to lack of comprehensive test data and lack of inexpensive data masking techniques. Synthetic data was generated, and errors were introduced at different rates to calculate linkage quality.

VI. RESULTS AND CONCLUSION

The optimized matching algorithm outperforms naïve comparison of encrypted Bloom filters, as shown in Table 1.

TABLE I. TIME TAKEN BY THE PROPOSED APPROCH VS NAÏVE VECTOR HOMOMORPHIC ENCRYPTION OF BLOOM FILTERS

Matrix size	**Time taken (s)**			
	Vector encryption	***Matrix encryption***	***Vector matching***	***Matrix Matching***
4*4	0.0626682	0.0482091	2.07659	0.0584722
8*8	0.125355	0.0412167	8.06762	0.0563123
16*16	0.252727	0.10382	32.1595	0.147115
32*32	0.502159	4.01244	128.199	5.84446

Real-world limitations were taken into consideration while designing a privacy-preserving patient matching solution using homomorphic encryption that significantly lowers computational costs and communication overheads. Future findings will include statistics for larger matrices, using key recryption to reduce the size of keys required.

ACKNOWLEDGMENT

This research was conducted by Brainlab AG, in collaboration with Beame IO towards building patient record registries in Germany.

REFERENCES

[1] P. Christen, R. Schnell, D. Vatsalan, and T. Ranbaduge, "Efficient cryptanalysis of bloom filters for privacy-preserving record linkage", Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, Cham, May 2017, pp. 628-640.

[2] S. M. Randall, A. P. Brown, A. M. Ferrante, J. H. Boyd, and J. B. Semmens, “Privacy preserving record linkage using homomorphic encryption”, Population Informatics for Big Data, Aug. 2015, 10.

[3] M. S. H. Cruz, T. Amagasa, C. Watanabe, W. Lu, and H. Kitagawa, "Secure similarity joins using fully homomorphic encryption", Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services, ACM, Dec. 2017, pp 224-233.

[4] D. H. Duong, P. K. Mishra, and M. Yasuda, “Efficient secure matrix multiplication over LWE-based homomorphic encryption”, Tatra Mountains mathematical publications, 67(1), Sep. 2016, pp. 69-83.