sciendo

Ahmed Salem*, Pascal Berrang, Mathias Humbert, and Michael Backes

Privacy-Preserving Similar Patient Queries for Combined Biomedical Data

Abstract: The decreasing costs of molecular profiling have fueled the biomedical research community with a plethora of new types of biomedical data, enabling a breakthrough towards more precise and personalized medicine. Naturally, the increasing availability of data also enables physicians to compare patients' data and treatments easily and to find similar patients in order to propose the optimal therapy. Such similar patient queries (SPQs) are of utmost importance to medical practice and will be relied upon in future health information exchange systems. While privacy-preserving solutions have been previously studied, those are limited to genomic data, ignoring the different newly available types of biomedical data.

In this paper, we propose new cryptographic techniques for finding similar patients in a privacy-preserving manner with various types of biomedical data, including genomic, epigenomic and transcriptomic data as well as their combination. We design protocols for two of the most common similarity metrics in biomedicine: the Euclidean distance and Pearson correlation coefficient. Moreover, unlike previous approaches, we account for the fact that certain locations contribute differently to a given disease or phenotype by allowing to limit the query to the relevant locations and to assign them different weights. Our protocols are specifically designed to be highly efficient in terms of communication and bandwidth, requiring only one or two rounds of communication and thus enabling scalable parallel queries. We rigorously prove our protocols to be secure based on cryptographic games and instantiate our technique with three of the most important types of biomedical data – namely DNA, microRNA expression, and DNA methylation. Our experimental results show that our protocols can compute a similarity query over a typical number of positions against a database of 1,000 patients in a few seconds. Finally, we propose and formalize strategies to mitigate the threat of malicious users or hospitals.

Keywords: Biomedical data privacy

DOI 10.2478/popets-2019-0004

Received 2018-05-31; revised 2018-09-15; accepted 2018-09-16.

1 Introduction

With the plummeting costs of molecular profiling, and in particular whole-genome sequencing, the amount of personal biomedical data available is rapidly increasing. This new data availability has positively affected the research in medicine by enabling the development of research directions that were previously impossible due to the lack of data. Whole-genome sequencing has enabled biomedical researchers to identify the relations between a plethora of severe diseases and the genomic mutations responsible for them. For instance, links have been found between some genomic variants and the risk of developing Alzheimer's disease [1]. However, the genome is only the tip of the iceberg concerning the current biomedical data deluge.

Researchers have shown that aberrant levels of other types of biomedical data, such as epigenomic or transcriptomic data, could indicate its owner carrying a severe disease, such as diabetes, neurodegenerative diseases, heart diseases or cancers [2–4]. Two of the most important epigenetic elements in the human body, DNA methylation and microRNAs (miRNAs), have been associated with numerous diseases as well. For example, abnormal DNA methylation patterns, such as hyper- or hypomethylation, are often observed for cancer patients, leading to the hyper-activation of genes such as oncogenes, or the silencing of tumor suppressor genes [5]. Dysregulation of microRNA expression has also been shown to be linked to several types of cancer [6]. Finally, the combination of different kinds of biomedical data is an auspicious and growing direction of research. For in-

***Corresponding Author: Ahmed Salem:** CISPA, Saarland University, E-mail: ahmed.salem@cispa.saarland
Pascal Berrang: CISPA, Saarland University, E-mail: pascal.berrang@cispa.saarland
Mathias Humbert: Swiss Data Science Center, ETH Zurich and EPFL, E-mail: mathias.humbert@epfl.ch
Michael Backes: CISPA Helmholtz Center i.G., E-mail: backes@cispa.saarland

stance, Hamed et al. study the relation between breast cancer and the combination of various data such as DNA methylation, microRNA expression, and genomic variants [7]. Speicher and Pfeiger combine DNA methylation, microRNA expression, and gene expression for integrative analysis of tumor samples [8, 9].

As biomedical data are increasingly available, it becomes easier to compare patients' data with each other. This functionality is of utmost importance to be able to find similar patients and, e.g., check how they responded to different therapies. In this context, the Global Alliance for Genomics and Health (GA4GH) has developed the MatchMaker Exchange [10], a platform that facilitates rare disease gene discovery through a federated network of genotype and phenotype databases. It typically answers requests such as "do you have any patients similar to one who has hypertelorism with a deleterious variant in CASQ2", where the similarity is defined by the receiving system. However, due to the serious privacy concerns related to genomic data [11–16], but also epigenomic and transcriptomic data [17–21], we anticipate that any similarity computation will have to be performed privately in the near future, i.e., without disclosing the patients' raw biomedical data.

While privacy-preserving similar patient queries have been previously studied in the literature, these solutions have been practically limited to whole-genome queries [22–24]. In this paper, we propose new methods that allow us to find similar patients based on various types of biomedical data, such as genomic, epigenomic, and transcriptomic data, and to combine them if necessary. Moreover, our approach easily allows the users to query specific locations instead of the whole genome. This is especially important as *omics*-based diseases and therapies are always related to a limited set of positions [25–30]. Moreover, it easily allows a medical practitioner to find similar patients carrying a certain disease by querying the relevant disease markers for the chosen set of biomedical data types. Finally, it is generic enough to adapt to the functionality of any health information exchange platform, such as the MatchMaker Exchange.

Contributions

In this paper, we introduce the first privacy-preserving similar patient query system that can handle different biomedical data types, while enabling the users to query a specific subset of positions and to put different weights on these positions or data types. Our system relies on additively homomorphic encryption and implements different protocols for performing similar patient queries based on two similarity measures widely used in the biomedical context: the Euclidean distance and the Pearson correlation coefficient.

Since the secure computation of Pearson correlation coefficient involves squaring of ciphertexts, we also propose an efficient extension for additively homomorphic encryption schemes that enables the squaring of ciphertexts. An additional requirement on the encryption scheme is that it also supports scalar multiplication of ciphertexts. Our extension is based on the work of Catalano and Fiore [31], adapting their idea to squaring and improving the efficiency in terms of ciphertext size and decryption time. We also prove our extension to be secure with regard to ciphertext indistinguishability under chosen plaintext attacks. We instantiate our protocols and the proposed extension with the Paillier cryptosystem.

We consider three different settings of information released to the parties involved in our protocols, thus offering some flexibility to the system designer. These range from both queried and querying parties learning only whether the similarity measure is below or above a predefined threshold (boolean outcome) to one of the two parties learning the final similarity measure and the other learning the boolean outcome. The latter case can be helpful when the querying party (e.g., a physician) wants to know to which extent his patient is similar to patients in the queried party's database (e.g., in a hospital's database). All our protocols are designed with a focus on keeping the communication overhead low, running in only one or two rounds of communication depending on the chosen setting. This allows for scalable parallel queries, effectively reducing the number of open connections and the bandwidth on the hospital's side.

We rigorously prove our protocols to be secure in the presence of honest-but-curious user and hospital. Moreover, we provide an in-depth discussion and formalize countermeasures to prevent inferences from malicious users or hospitals. Finally, we implement and test the different protocols using various combinations of data. These show that we can query a typical number of biomedical data positions against a database of 1,000 patients in less than five seconds for the weighted Euclidean distance protocol, and in less than thirty seconds for the Pearson correlation coefficient. This demonstrates that our approach is efficient and highly practical in current clinical and research settings.

2 Background

2.1 Similar Patient Query

A similar patient query (SPQ) denotes a query carried out by a user (e.g., a medical practitioner) on a set of patients (usually from a hospital database). The query takes a patient's data as input and returns a set of patients in the hospital database similar to the input patient. A single comparison between two patients can also be referred to as an SPQ. SPQs are crucial for medical practice. They are used, e.g., when a physician needs to find similar patients to his own in order to study their treatment history. The resulting information can then be used to optimize the treatment for his own patient. As biomedical data is privacy sensitive, it is crucial to protect both the patient's data and the hospital's data.

While the previously mentioned use case is essential in the clinical environment, SPQs are not limited to this setting. Such queries can also be employed to tell whether participants of a study conducted by multiple research groups are similar to each other or not. Often, it is also desirable in these cases to protect the datasets of each laboratory. A third scenario to mention is the use of SPQs for consumer-oriented portals such as Ancestry [32] or 23andMe [33] offering to find relatives based on the similarities within the genome. Two of the most popular similarity measures used in biomedical practice are the Euclidean distance and Pearson correlation coefficient [34–38]. We provide protocols for performing SPQs *securely* based on the two aforementioned similarity measures in a way that none of the participants learns the other's data.

2.2 Biomedical Data

While there exist a plethora of different types of biomedical data, we focus on three of the most common types: genomic data, microRNA expression profiles, and DNA methylation profiles. Our approach, however, is general enough to capture any biomedical data that can be represented by an integer or a real value.

2.2.1 Genome

The first data type we consider is the genome. Abstractly, the genome is a pair of sequences of three billion bases taking value in $\{\mathsf{A},\mathsf{T},\mathsf{G},\mathsf{C}\}$. This pair of sequences carries the genetic signature of its holder.

The genomes of two human beings only differ in around 0.1% of the positions. These locations varying between individuals are called single nucleotide polymorphisms (SNPs). There are currently about 150 million referenced SNPs on dbSNP[39]. At each SNP location within a single DNA sequence, there are two possible nucleotide values. The most frequent value is called the major allele, and the other the minor allele.

As the DNA is a pair of sequences, at any SNP position there are two alleles. Hence, there are only three possible combinations for the values of a SNP which are: (i) both are major alleles, (ii) one is major and the other is minor, or (iii) both are minor alleles. We usually encode these combinations as $0, 1$ and 2, respectively. Therefore, a genome of length L_1 can be encoded as $\{0,1,2\}^{L_1}$. Note that we are interested in a subset of length $L'_1 \ll L_1$ in our SPQ setting. We describe how the genomic data comparison can be combined with the other biomedical data types in Section 6.1.

2.2.2 MicroRNA Expression

The second type of biomedical data is microRNA (miRNA) expressions data. MiRNAs are small non-coding RNA molecules containing about 22 nucleotides and are mainly responsible for gene regulation, i.e., which parts of the DNA are active in a given cell. There are more than 5,000 known miRNAs. The expression of microRNA is proportional to the number of active miRNA molecules expressed in a group of cells and thus is represented by a positive, real value. A microRNA expression profile of size L_2 can be formally represented as $\mathbb{R}_+^{L_2}$. As for the genome, we are interested in a subset of the microRNA expression points of size $L'_2 \ll L_2$. In Section 6.1, we explain how we encode these values as fixed point integers in our implementation.

2.2.3 DNA Methylation

The third type of biomedical data we consider is DNA methylation. In DNA methylation, a special molecule is attached to a certain position of the DNA. Specifically, such a methylation can only occur at positions where a C nucleotide is followed by a G. While a position in a single cell can only be methylated or not, the DNA methylation can vary across different cells and copies of the DNA. Hence, DNA methylation is measured as a real value between 0 and 1, which represents the fraction of cells that are methylated at a particular position. There are around 28 million known DNA methylation positions in the human genome. A DNA methylation profile of size L_3 is formally structured as $[0,1]^{L_3}$. Again, for our SPQ, we are only interested in a subset of the methylation positions of size $L'_3 \ll L_3$. As for the microRNA expression, we describe how we encode the methylation values as fixed point integers in Section 6.1.

3 System Model

Our protocols assume two participating parties we refer to as the hospital and the user. The hospital is the entity that possesses and manages the database of the biomedical data. In principle, there can be multiple hospitals. Although we name this party a hospital, it could as well be any organization that collects and stores biomedical data. The second party in our protocols is the user, who initiates the query. The user is interested in performing an SPQ against a patient in the hospital database.

We aim to compute the similarity between two patients. One patient's data is known to the hospital, and the other patient's data is known to the user. None of the parties should learn the other party's data. Regarding the outcome of the SPQ, we consider three different scenarios of information released to either party. In the first setting, the hospital is allowed to learn the similarity measure outcome and returns to the user only whether this outcome is above or below a predefined threshold. In the second setting, both parties only learn whether the similarity measure outcome is above or below the threshold. Lastly, in the third setting, the hospital does not learn anything, and the user learns the similarity measure outcome.

In our approach, we assume an honest-but-curious user and hospital, i.e., they have to follow the protocol but may try to extract or infer more information from any data they can get. The honest-but-curious hospital and user model is widely accepted in the literature [20, 22, 40–42]. However, we propose extensions for the defense against a fully malicious user and hospital in Section 9 to show the generalizability of our approach for different use cases.

4 Building Blocks

In this section, we first recall the similarity measures we use in our protocols. Then, we present the basics of public key encryption and our extension to additively homomorphic encryption for squaring operations.

4.1 Similarity Measures

In this paper, we focus on two types of similarity metrics, namely Pearson correlation coefficient (PC) and the Euclidean distance (ED) since both metrics are frequently used in the context of biomedical applications. However, our general scheme can be easily extended to other similarity measures, such as cosine similarity.

PC measures the linear correlation between two sets of values. The coefficient's value is between $+1$ and -1, where $+1$ means complete positive correlation and -1 means complete negative correlation. We use the following equation for calculating the PC:

$$\frac{n\sum_{i\in I} x_i y_i - \sum_{i\in I} x_i \sum_{i\in I} y_i}{\sqrt{(n\sum_{i\in I} x_i^2 - (\sum_{i\in I} x_i)^2)(n\sum_{i\in I} y_i^2 - (\sum_{i\in I} y_i)^2)}} \tag{1}$$

where I is the set of positions the user is interested in, x and y are the two vectors whose similarity he wants to measure.

ED is always greater than or equal to zero, and the smaller the distance, the more similar two patients are. We use the following equation for estimating ED:

$$\sum_{i\in I}(x_i - y_i)^2 \tag{2}$$

In fact, the actual ED is the square root of that equation. However, since we just compare the distance with a threshold, we can square both the threshold and the real ED to facilitate the computation under encryption and end up with the same result.

4.2 Public-key Encryption

Public-key encryption was first introduced by Diffie and Hellman in [43], and numerous schemes for public-key cryptography have been proposed since then. In our paper, we require a public-key encryption scheme with the additively homomorphic property, allowing to add messages under encryption. More formally, we need a scheme $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Smul})$ such that:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$, the generation protocol generates a pair of public and private keys on the input of the security parameter.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, the encryption protocol outputs a ciphertext on the input of a message and the public key.
- $\{m, \perp\} \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$, the decryption protocol outputs a message on the input of a valid ciphertext and a private key, otherwise it fails.
- $c_2 \leftarrow \mathsf{Add}(c, c_1)$, the addition protocol, which outputs a ciphertext on the input of two ciphertexts. The output ciphertext holds the sum of m and m_1 where m and m_1 are the decryptions of c and c_1.
- $c_2 \leftarrow \mathsf{Smul}(k, c_1)$, the scalar multiplication protocol, which outputs a ciphertext on the input of a scalar and a ciphertext. The output ciphertext contains the multiplication of k and m_1 where m_1 is the decryption of c_1.

There are many available schemes fulfilling our requirement. We will rely on the Paillier cryptosystem [44] for our implementation and evaluation.

4.3 Squaring of an Encrypted Message

Since both our similarity measures in Section 4.1 require the user to calculate the square of an encrypted message, we propose an extension of any additively homomorphic encryption scheme supporting scalar multiplication. We will instantiate our extension exemplarily for the original Paillier cryptosystem providing the ability for the squaring operation. While plain additive homomorphic encryption does not support such an operation, we follow the approach of Catalano and Fiore [31], which extends an additively homomorphic encryption scheme with a multiplication operation. However, the primary drawback of their scheme is that the multiplication operation and its subsequent operations can substantially increase the size of the ciphertext.

Therefore, we adapt their scheme to only support squaring of encrypted messages. Our modification eliminates one component of the original construction and thus can significantly reduce the ciphertext size, thereby also decreasing the decryption time by almost a factor of two. We build on an additive homomorphic encryption scheme $\hat{\pi} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Add}', \mathsf{Smul}')$ over the message space $\mathcal{M}$. Where Gen' is the key generation algorithm, Enc' is the encryption algorithm, Dec' is the decryption algorithm, Add' is the ciphertext addition algorithm, and Smul' is the scalar multiplication algorithm. Leveraging this scheme, we build our new scheme $\pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over the same message space with the operations

$$\{\mathsf{Add}(c_1, c_2), \mathsf{Smul}(k, c), \mathsf{Square}(c)\}$$

In the following, we will define the operations on our scheme π, starting with the encryption.

4.3.1 Encryption

In principle, now every ciphertext is represented by a pair $c = (\alpha, B)$, with α being a ciphertext of the underlying scheme $\hat{\pi}$ and B being a list of ciphertexts β_i of $\hat{\pi}$.

A freshly created ciphertext has an empty list B and the message m is encrypted with the underlying scheme $\hat{\pi}$ and stored in the first part of the ciphertext α, i.e., $\mathsf{Enc}(m) = (\mathsf{Enc}'(m), [\,])$ with $[\,]$ being the empty list.

4.3.2 Decryption

The decryption of a ciphertext $\mathsf{c} = (\alpha, \mathsf{B})$ works as follows:

$$\mathsf{Dec}((\alpha, B)) = \mathsf{Dec}'(\alpha) + \sum_{\beta_i \in B} \mathsf{Dec}'(\beta_i)^2.$$

4.3.3 Addition

To add two ciphertexts $c_1 = (\alpha_1, B_1)$ and $c_2 = (\alpha_2, B_2)$ the α's are just added together using the homomorphic property and the B's are concatenated. Hence, we get

$$\mathsf{Add}(c_1, c_2) = (\mathsf{Add}'(\alpha_1, \alpha_2), B_1 || B_2),$$

where $||$ is the list concatenation.

4.3.4 Scalar Multiplication

To scale a ciphertext c by a factor k, we distinguish between two cases. The first is when c is a simple ciphertext, i.e., ciphertexts for which the list B is empty. In this case, $\mathsf{Smul}(k, c) = \mathsf{Smul}(k, (\alpha, [\ ])) = (\mathsf{Smul}'(k, \alpha), [\,])$. The second case is when c has a list B which is not empty. In this case, we only support scalar multiplications with factors which have integer square roots. To scale a ciphertext c with a factor k, we scale α with k, and then scale all the ciphertexts in the list B

with the square root of k. More formally, $\mathsf{Smul}(k,c) = \mathsf{Smul}(k,(\alpha,B)) = (\mathsf{Smul}'(k,\alpha), [\mathsf{Smul}'(\sqrt{k},\beta_i)]_{\beta_i\in B})$.

4.3.5 Squaring

Below is the modified multiplication protocol from [31], adapted only to calculate the square $\mathsf{Square}(c)$ of the ciphertext c. The following protocol does only rely on the given operations on ciphertexts and can hence be executed without the knowledge of m.

1. Let $\mathsf{Enc}(m) = c = (\alpha', B')$. If B' is not empty, throw an error since we only support a single squaring operation.
2. Pick a random value $a \leftarrow \mathcal{M}$.
3. Calculate $\gamma_1 = \mathsf{Enc}'(-a)$.
4. Calculate $\beta = \mathsf{Add}'(\alpha', \gamma_1) = \mathsf{Enc}'(m - a)$.
5. Calculate $\gamma_2 = \mathsf{Enc}'(-a^2)$.
6. Calculate $\gamma_3 = \mathsf{Smul}'(2a, \alpha') = \mathsf{Enc}'(2am)$.
7. Calculate $\alpha = \mathsf{Add}'(\gamma_2, \gamma_3) = \mathsf{Enc}'(-a^2 + 2am)$.
8. Return $(\alpha, [\beta])$, where $[\beta]$ denotes the list with only β as element.

Compared to the original multiplication protocol, our modification reduces the number of random values to one instead of two, effectively decreasing the size of β by a factor of 2. β and α can be calculated efficiently because of the additive and scaling properties of the encryption scheme. The additive property allows the addition of the messages under encryption. The scaling property allows the ciphertexts to be scaled by a scalar.

The correctness of these modifications follows from the following theorem.

Theorem 1. *The squaring modification is correct, i.e.,* $\mathsf{Dec}(\mathsf{Square}(\mathsf{Enc}(m))) = m^2$

Proof.

$$\mathsf{Dec}(\alpha) = -a^2 + 2am \tag{3}$$
$$\mathsf{Dec}(\beta)^2 = (m - a)^2 \tag{4}$$

Simplifying Equation 4 results in

$$\mathsf{Dec}(\beta)^2 = m^2 - 2am + a^2 \tag{5}$$

Adding Equation 3 and 5 results in

$$-a^2 + 2am + m^2 - 2am + a^2 = m^2 \tag{6}$$

□

The security proof is given in Appendix B.

5 Assumptions

We rely on the following two assumptions for our system. First, we assume there exists a scheme $\pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Smul})$, which is a CPA secure additively homomorphic public encryption scheme . Gen is the key generation algorithm, Enc is the encryption algorithm, Dec is the decryption algorithm, Add is the ciphertext adding algorithm, and Smul is the scalar multiplication algorithm. Second, we assume the non-collusion between users and hospital. The non-collusion assumption makes sense in our case because, if they collude, then no information needs to be hidden, as they can directly share their data with each other. In a realistic scenario, the hospital would try to learn the user's data, and the user would try to learn the hospital's data.

6 Design

In this section, we present our protocols for performing similar patient queries based on diverse biomedical data.

6.1 Setup

As previously mentioned, genomic data are usually represented as a sequence of nucleotides or, if we focus on SNPs, as an integer between 0 and 2. MicroRNA expression and DNA methylation data are usually represented in the form of real values, with DNA methylation being bounded between 0 and 1.

Since the different representations of data can be a barrier to combine all of them for calculating the SPQ, we unify their representation by encoding them as fixed point integers. We consider SNPs to be represented by 0, 1, or 2. DNA methylation values and microRNAs are scaled up by a factor of 10^p for p digits of precision and then rounded to the nearest integer. In this work, we set $p = 8$ digits of precision for DNA methylation values because the methylation values in our dataset are encoded with 8 digits. For the same reason, we set $p = 9$ digits of precision for the microRNA expression values.

Since all data are now presented as integers, the hospital publishes all of its data under encryption and reveals the locations and types of the data. A user then picks the positions he is interested in and calculates the similarity value, which he will send to the hospital in order to obtain the final result.

Inputs of User:	**Hospital's encrypted values, Hospital's encrypted squared values, Hospital's public key, his patient data, weights and set I of the positions he is interested in.**
Inputs of Hospital:	**Private key and threshold.**
Output:	**If ED between the hospital's and user's data is below the threshold or not.**
User:	**Encrypts the data variants he wants to query and their squared values.**
User:	**For each $i \in I$ he calculates $r_{1i} = \mathsf{Enc}(h_i^2) \cdot \mathsf{Enc}(u_i^2)$ which is equal to $\mathsf{Enc}(h_i^2 + u_i^2)$.**
User:	**For each $i \in I$ he calculates $r_{2i} = \mathsf{Enc}(h_i)^{(-2u_i)}$ which is equal to $\mathsf{Enc}(-2h_i u_i)$.**
User:	**Calculates $res = \sum_{i \in I} r_{1i} \cdot r_{2i}$ which is equal to $\sum_{i \in I} \mathsf{Enc}(h_i^2 + u_i^2 - 2h_i u_i)$. The user can apply different weights for every position by just multiplying the weights to the ciphertexts in each row using this equation $\sum_{i \in I} \mathsf{Enc}(h_i^2 + u_i^2 - 2h_i u_i)^{w_i}$.**
User $\rightarrow$ Hospital:	**User sends res to the hospital.**
Hospital:	**Decrypts res.**
Hospital $\rightarrow$ User:	**Hospital checks the value against some threshold which is defined by the hospital. If res is below this threshold then the hospital sends to the user that these are similar else the hospital sends that they are not similar. This can be even fine tuned so instead of sending just true or false the hospital can send how similar this user is in a scale for example from 1 to 10.**

Fig. 1. Our protocol for Similar Patient Queries using Euclidean distance.

It is important to mention that we are using additively homomorphic encryption scheme, hence the user is able to apply different weights with different positions or data types. He can apply the weights both to his values and to the hospital's without any interaction with the hospital. This is particularly crucial to account for the scaling performed to the data types. For instance, if the user wants to cancel the scaling factor while using the PC version of the scheme, he just needs to make all values scaled with the same factor. The scaling factor will be canceled without any extra actions, due to the PC Equation 1. For the ED version, the user would need to divide the final value with the square of the scaling factor to cancel it. An example for scaling the values would be to multiply the methylation values by 10, and DNA values by 10^9 to make them 9 digits similar to the microRNA expression values.

We assume the hospital to create a key pair $(\mathsf{pk}, \mathsf{sk})$, which is the key for the encryption scheme. For our evaluation, we rely on the adapted version of the Paillier cryptosystem. Using this key pair, the hospital publishes the patients' biomedical data, including the positions, the type of data, the encryption of the data itself. We further assume the hospital to publish the encryption of the square of the patients' data values. While the square could in principle be calculated by the user, directly publishing it is much more efficient.

For each of the similarity measures used in this paper, we will present three settings differing in the information release to each party. Table 1 summarizes the three settings we consider. A "boolean" information release to a party means that this party only learns whether the similarity is below or above a predefined threshold, whereas releasing the "similarity" means the party learns the actual outcome of the similarity computation.

Table 1. Three settings considered in our SPQ protocols with regard to the information release to each party.

Setting	Hospital	User
1)	similarity	boolean
2)	boolean	boolean
3)	nothing	similarity

6.2 SPQ using Euclidean Distance

Scheme 1 presents our protocol for calculating the SPQ using ED in the basic setting, i.e., when the hospital does learn the similarity value and returns to the user whether it is below or above the predefined threshold. However, instead of considering the equation for ED as presented in Section 2, we will use an expanded version of it as follows:

$$\sum_{i \in I} x_i^2 - 2x_i y_i + y_i^2$$

By using the expanded version and the additionally published squared values, we can avoid calculating the square of a ciphertext, which is not only computationally expensive but results in increasing the ciphertext size as mentioned in Section 4.3. The only disadvantage of using that equation is that now the hospital needs to also publish the encryption of the square of the values. However, this is already required for the SPQ using PC as will be explained in Section 6.3.

As a reminder, I is the set of positions the user is interested in. Let h be the vector containing the hospital's patient's data, u be the vector containing the user's patient's data and w be the vector containing the values of the user's weight for each position of the set I.

Scheme 1 also explains how to incorporate the weights for individual positions during the computation. Basically, the user assigns different weights for every position by multiplying them to the ciphertexts in each row following the equation $\sum_{i \in I} \mathsf{Enc}(h_i^2 + u_i^2 - 2h_i u_i)^{w_i}$.

In our second setting, we hide the similarity value from the hospital and only let both parties learn whether it is above or below the threshold. This can be easily achieved by leveraging the ciphertext comparison protocol proposed in [45]. The hospital would publish the encryption of the threshold and the user compares it to his result using the building block, sending only the comparison result to the hospital instead of res.

Our third setting, in which only the user learns the similarity value, can be easily achieved by the user blinding res. The hospital then sends the decrypted value to the user, who unblinds it and hence receives the desired outcome.

6.3 SPQ using Pearson Correlation Coefficient

Scheme 2 presents our protocol for calculating the SPQ using PC in the basic setting, in which the hospital learns the correlation coefficient and returns to the user only whether it is below or above a threshold. I is the set of positions the user is interested in, h the vector containing the hospital's patient's data, and u be the vector containing the user's patient's data.

As shown in Scheme 2, we use our ciphertext squaring extension to calculate p_7. According to the Equation 1, p_7 is supposed to be multiplied by -1. However, our ciphertext squaring extension presented in Section 4.3 does not support such multiplication. In order to bypass this issue, we calculate the opposite (negative value) of both parts of the denominator in (1) leading to the same final value since the two negative values cancel out.

For the blinding and unblinding of p_{10} and p_{11}, it is important to note that the blinding and unblinding are performed in different groups. The blinding happens on the encrypted values, but the unblinding happens on the decrypted values. To ensure a correct unblinding, overflow of the blinded values should be avoided. This is not an issue in our scenario as the considered values are significantly smaller than the modulus size. More concretely, our data have maximum 9 digits which can be encoded in 30 bits, which is much smaller than the size of the cryptosystem modulus (2048 bits) that we use in Section 8. Finally, there is no constraint on r_h as the blinding and unblinding operations are performed in plain.

In order to hide the resulting correlation coefficient from both parties, corresponding to our second setting, and to only reveal whether the result is above or below the hospital's threshold, the hospital scales and encrypts p_{14} before sending it to the user. The scaling depends on the accuracy needed by the hospital and the threshold should also be scaled with the same scaling factor, which should be known and public. Then, the hospital sends this encryption to the user who removes the blinding. Finally, both parties run the ciphertext comparison protocol proposed in [45] on the scaled threshold and the unblinded encryption.

For our third setting, the protocol can be adapted to let the user know the result instead of the hospital. This is achieved by the hospital not blinding p_{12}, which enables the user to learn the correlation coefficient.

6.4 Advantages of our Approach

Here we highlight the main advantages of our approach, notably with respect to previous contributions.

First of all, while some related protocols require a trusted third party such as the one presented in [24], we entirely eliminate the need thereof because neither the security nor the functionality of our protocols depend on any external parties.

Moreover, to the best of our knowledge, we are the first to provide and evaluate protocols that are explicitly able to combine different types of biomedical data, also considering different weights for specific positions. They are essential to give specific positions or biomedical data more weight for the similarity calculation. Moreover, weights can also be used to give the user the flexibility of normalizing the distance if needed.

Compared to most other approaches, our protocols only require a very low amount of communication. One

Inputs of User:	**Hospital's encrypted values, Hospital's encrypted squared values, Hospital's public key, his patient data and set I of the positions he is interested in.**
Inputs of Hospital:	**Private key and threshold**
Output:	**If the PC between the hospital's and user's data is below the threshold or not**
User:	**Calculates $\sum_{i \in I} u_i$. Then multiplies the encryption of the hospital's values to get $\mathsf{Enc}(\sum_{i\in I} h_i)$ and raises it to the previously calculated $\sum_{i \in I} u_i$ to get $p_1 = \mathsf{Enc}(\sum_{i\in I} u_i \sum_{i\in I} h_i)$.**
User:	**For every i the user raises $\mathsf{Enc}(h_i)$ to u_i to get $\mathsf{Enc}(h_i u_i)$ then multiplies these ciphertexts to get $\mathsf{Enc}(\sum_{i\in I} h_i u_i)$. Finally, he raises it to n, where n is the length of I to get $p_2 = \mathsf{Enc}(n \sum_{i\in I} h_i u_i)$.**
User:	**multiplies p_1^{-1} with p_2 to get $p_3 = \mathsf{Enc}(n\sum_{i\in I} h_i u_i - \sum_{i\in I} h_i \sum_{i\in I} u_i)$.**
User:	**Calculates $p_4 = (\sum_{i\in I} u_i)^2 - n\sum_{i\in I} u_i^2$.**
User:	**Multiplies the encryption of the hospital's squared values to get $\mathsf{Enc}(\sum_{i\in I} h_i^2)$, then raises it to n to get $p_5 = \mathsf{Enc}(n\sum_{i\in I} h_i^2)$.**
User:	**Multiplies the encryptions of the hospital's values to get $p_6 = \mathsf{Enc}(\sum_{i\in I} h_i)$ then squares it to get $p_7 = \mathsf{Enc}((\sum_{i\in I} h_i)^2)$. This squaring is done with the extension we previously introduced in 4.3.**
User:	**Multiplies p_5^{-1} to p_7 to get $p_8 = \mathsf{Enc}((\sum_{i\in I} h_i)^2 - n\sum_{i\in I} h_i^2)$.**
User:	**Raises p_8 to the square of p_4, to get $p_9 = \mathsf{Enc}(((\sum_{i\in I} h_i)^2 - n\sum_{i\in I} h_i^2)((\sum_{i\in I} u_i)^2 - n\sum_{i\in I} u_i^2)^2)$.**
User:	**Generates r_1 and r_2 as specified in Subsection 6.3**
User:	**Blinds p_9 with the square of a random number r_1 to get $p_{10} = \mathsf{Enc}({r_1}^2((\sum_{i\in I} h_i)^2 - n\sum_{i\in I} h_i^2)((\sum_{i\in I} u_i)^2 - n\sum_{i\in I} u_i^2)^2)$.**
User:	**Blinds p_3 with the a random number r_2 to get $p_{11} = \mathsf{Enc}(r_2(n\sum_{i\in I} h_i u_i - \sum_{i\in I} h_i \sum_{i\in I} u_i))$.**
User $\rightarrow$ Hospital:	**User sends p_{10} and p_{11} to the hospital.**
Hospital:	**Decrypts p_{10}, and calculates the square root of it. Then divides it with the decryption of p_{11} to get $p_{12} = \frac{r_2(n\sum_{i\in I} h_i u_i - \sum_{i\in I} h_i \sum_{i\in I} u_i)}{r_1(\sqrt{(n\sum_{i\in I} h_i^2 - (\sum_{i\in I} h_i)^2)(n\sum_{i\in I} u_i^2 - (\sum_{i\in I} u_i)^2)^2})}$.**
Hospital $\rightarrow$ User:	**Hospital picks a random r_h and sends $p_{13} = p_{12} \cdot r_h$ to the user.**
User:	**The user removes his blinding by multiplying p_{13} with $\frac{r_1}{r_2}$, then multiplies it with p_4 to remove the extra scaling and get p_{14}.**
User $\rightarrow$ Hospital:	**User sends p_{14} to the hospital.**
Hospital:	**The hospital removes its blinding by dividing p_{14} with r_h to get the coefficient and decides if the patients are similar or not.**
Hospital $\rightarrow$ User:	**Sends the user if the patients are similar or not.**

Fig. 2. Our protocol for Similar Patient Queries using Pearson Correlation Coefficient.

or two rounds of communication suffice to obtain the final results depending on which variant is used. This is important to reduce the online time of both parties. In other words, the low number of rounds avoids forcing both parties to be online to start the SPQ and stay online during the computation. Furthermore, a low amount of communication reduces the amount of bandwidth needed for each query. This is not only beneficial when limited bandwidth is available, but also when a hospital receives many queries at the same time.

Moreover, our protocols enable the user to query certain positions without the hospital knowing the positions. Since the hospital publishes the list of encrypted values for all positions, the user is free to compute the similarity on any set of positions without the hospital learning this set. We also discuss an extension of our protocols allowing to hide the result of the similarity computation from the hospital and instead only revealing whether the value is above or below a threshold. Finally, our protocol is generic enough to also apply to other similarity measures, data types and use cases such as the one presented in Appendix A.

7 Security and Correctness Analysis

In this section, we analyze and prove the security and correctness of our building blocks and protocols.

7.1 Correctness

The correctness property is defined as: the two parties engaged in the scheme should calculate the desired similarity metric. It is relatively straightforward to see the correctness of the scheme calculating ED (cf. Section 6.2) and PC (cf. Section 6.3).

For ED scheme, the correctness follows from that we just calculate the equation but under encryption, which does not affect the final output. We use the expanded version of ED as presented in Equation 2 and compare it with the square of the threshold.

The scheme calculating PC uses the exact equation for the coefficient as presented in Equation 1. The blinding factors added in the scheme are all removed at a later stage, thus eventually providing the exact PC.

7.2 Security

Due to space constraints, we present the definitions and proofs related to the following theorems in Appendix B.

Theorem 2. *Let Π_{PKE} be IND-CPA secure. Then the squaring modification is CPA secure.*

Theorem 3. *Let Π_{PKE} be IND-CPA secure. Then the SPQ using ED protocol is secure.*

Theorem 4. *Let Π_{PKE} be IND-CPA secure. Then the SPQ using PC protocol is secure.*

8 Performance Evaluation

In this section, we first describe the three datasets we rely upon for our experiments and then present our experimental setup and results.

8.1 Datasets Description

In our experiments, we use three types of data, DNA, DNA methylation, and microRNA expression taken from three different datasets. For the genomic data, we use the publicly available data from the Personal Genomes Project (PGP) platform, which contains 7840 whole-genome sequences[46]. The data are stored in VCF (variant call format) files, which is a standard format in bioinformatics. We randomly extract from the VCF files 1000 SNP values. These values are 0, 1, and 2 as already mentioned depending on the alleles the SNP contains. The values 0, 1, and 2 are not equally distributed, the value 0 is the most frequent one. For the DNA methylation data, we rely on an epigenomic dataset from the Gene Expression Omnibus (GEO)[47] available under accession number GSE44684 [48]. We extract from the files the locations and methylation values of 67 individuals. Finally, for the microRNA expression data, we use the dataset under accession number GSE68951 from the Gene Expression Omnibus (GEO) [49]. We extract from the files the locations and the expression values of 26 patients. These three datasets show the general applicability of our approach to any type of biomedical data as well as their combination.

8.2 Experiments and Results

We run the evaluation on two different servers located in different states. The hospital is run on the Amazon AWS service with an instance of type c3.8xlarge, which has 60 GB of memory, 32 Intel Xeon E5-2680v2 cores, and a network bandwidth up to 10 Gbps. The user is run on a local server with a 768 GB of memory and 4 Intel Xeon E5-4650L octa-core processors, providing 32 physical CPUs, and has the Intel's hyper-threading technology enabled. Note that our protocol does not fully use the complete computational power of these machines. In practice, the maximum memory consumption of our protocols is about 20% of the hospital machine – 12 GB – and 2% of the user machine – 15 GB –, and the actual bandwidth of the network is about 200 Mbps. Both of the protocols use the second setting presented in Table 1 and are implemented in Java using multi-threading. As previously mentioned, we implement our evaluation instantiated by the Paillier cryptosystem [44] with a key size of 2048. It is also important to note that we implement the Paillier cryptosystem without any efficiency improvements, like those presented in [50]. These could be further implemented to reduce the running time.

We test different combinations of the number of participants, the number of positions, the data distribution with the protocol for the second setting, and every similarity measure variant. Recall that the second setting only reveals the boolean outcome to both parties. We chose to evaluate this setting, since it requires the most rounds of communication, thus providing an upper bound on our protocols. The first protocol we evaluate calculates ED without weights, the second protocol computes PC, and finally, the third protocol calculates the weighted Euclidean distance (WED). We randomly assign weights to every position independently. Regarding the biomedical data distributions we test the following combinations: We first assume that each type of data occupies 100% of the positions. Then, we assume one scenario in which each type of data takes an (almost) equal share in the similarity protocol, i.e., DNA (D) taking 33%, microRNA expression (R) 33% and DNA methylation (M) 34%.

We consider between 10 and 1000 positions in one of the three biomedical data types or spread among all of them if we assume a given disease is influenced by multiple types of data. In the latter case, we assume all three data types have an (almost) equal number of positions that contribute to the disease. For instance, if we consider 10 positions, we pick 3 SNPs from the genome, 3 microRNA expressions, and 4 methylation positions. We select a range between 10 and 1000 positions to be as close as possible to realistic queries. For example, many diseases are associated with less than 10 genomic positions [25–27], while other diseases are influenced by up to 100 positions [28, 29], and finally there are a few examples where we need up to 1000 positions, like the study on classifying brain tumors based on methylation profiles in [30]. We test the range from 100 to 1000 positions (with an increasing step of 100) especially for demonstrating the scalability of our approach and cover even rare cases where more than 100 positions are relevant.

Regarding the number of participants, we consider numbers ranging from 100 to 1000 participants in the hospital's database, with an increasing step of 100. Fig. 3 depicts the hospital's computational time and the total time including the communication time and the user's computational time for six different representative scenarios. First, we observe that the hospital computational time is almost constant regardless of the number of positions. This is because the hospital receives only one ciphertext no matter what the number of positions is. It is also independent of the number of participants because the hospital can operate on each participant's data separately, in other words, the hospital processes all participants' queries in parallel. Comparing the hospital time of Fig. 3, we observe that the data types that are used do not influence the hospital's computational time. This is expected as the hospital always gets the same formatted ciphertext, regardless of the type of data being encrypted.

We also observe that the hospital's time is almost equal for protocols WED and ED. This is not surprising because, in both cases, the hospital performs the same operations, as noted in Scheme 1. For the PC protocol, the hospital's time is more than four or five times the corresponding time in the other protocols. This increment is due to the hospital having to decrypt and work on more ciphertexts than for the other protocols as noted by the number of operations the hospital performs in Scheme 2 compared to the number of operations in Scheme 1. Furthermore, one of these ciphertexts is the result of the squaring of a number which is expected to take more time as described in Section 4.3. Finally, we observe that the hospital's computational burden is negligible compared to the user's as well as to the communication time in our protocols shown by the blue curves with square dots in the figures. The reason for this is that the user performs the computations himself to hide the locations he is interested in, and the hospital mostly just decrypts the resulting ciphertexts.

Looking at the total time of Fig. 3 (a)-(c), we observe that the running time is larger for methylation data than for genomic data. It is slightly shorter than with methylation data when we combine all three data types. This shows that considering or combining different biomedical data other than solely genomic data increases the complexity of same patient queries. In general, the DNA-based similarity test takes less than 30 seconds for 1000 users and 1000 positions, and less than 5 seconds for 100 positions. Methylation-based similarity tests take around 10 seconds for 1000 participants and 100 positions. Note that the results for microRNA data are almost the same as for DNA methylation data as expected due to the similar real-valued nature and pre-processing of these data.

We further observe that the total time is in general not only linear in the number of positions that are queried but also in the number of participants in the database. The only exception for this observation is Fig. 3b. This exception results from having a higher concentration of 0s in the first 500 SNPs than in the second 500 SNPs. The non-linear behavior is because addition of zeros is faster than the addition of the non-zero values. This is the case since we add a plaintext value – the user value – directly to a ciphertext – the hospital value –, not a ciphertext to a ciphertext, for efficiency reasons. Finally, comparing the different protocols, we notice first that ED protocol (Fig. 3f) is the fastest one, about two times faster than WED protocol (see Fig. 3c for comparison with 3f). Furthermore, PC protocol takes the longest time: it is about six times slower than WED protocol (see Fig. 3d and 3e for comparison). This is because, in the PC protocol, the user calculates much more than in the other protocols, as can be seen when comparing Scheme 2 with Scheme 1. Moreover, in the PC protocol, there is an additional round of communication that is needed in the middle of the protocol. The increment in time between protocols ED and WED is caused by the additional operations needed for applying the weights (cf. Scheme 1).

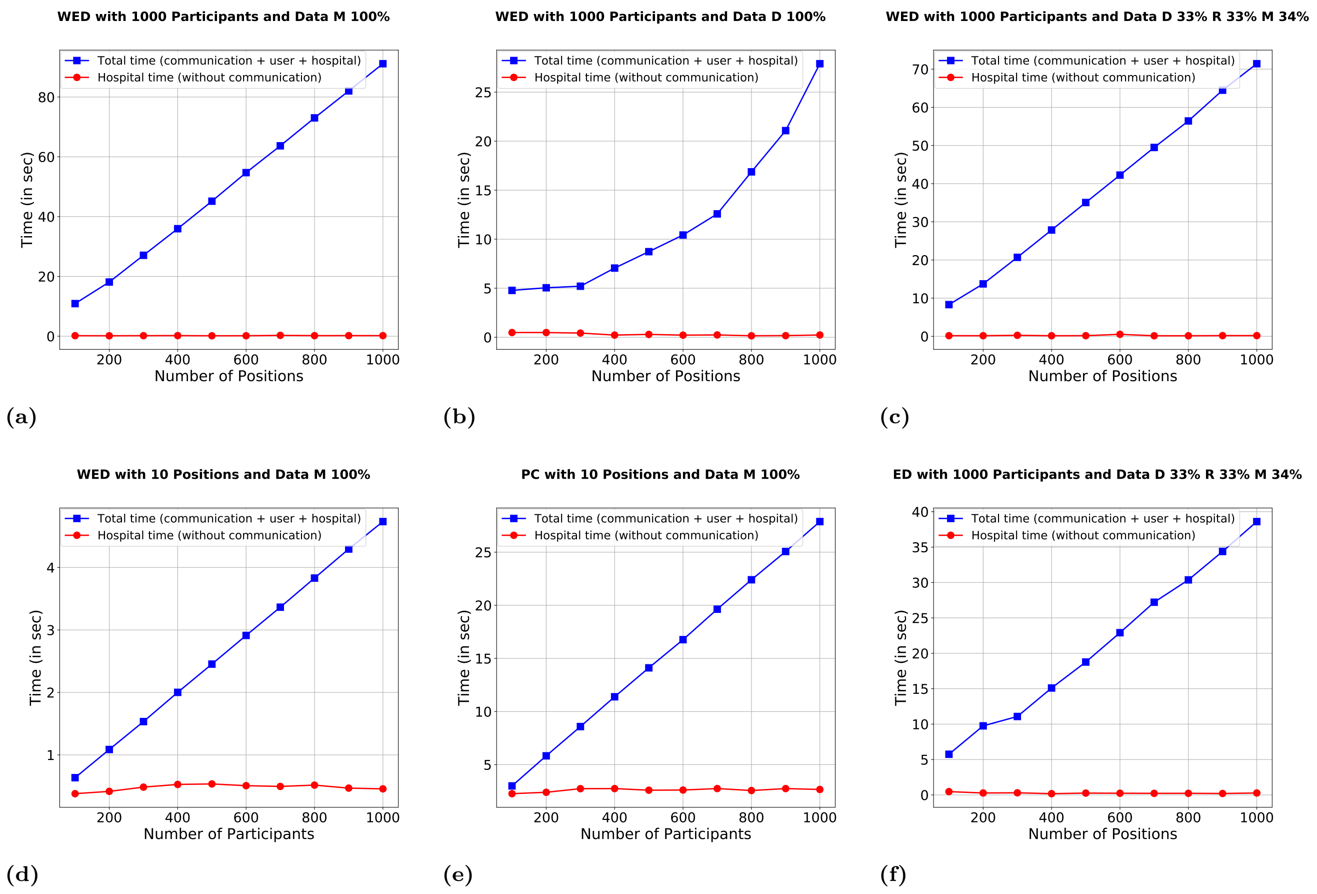

Fig. 3. Time performance of the different protocols under various data distribution, numbers of participants/positions scenarios. (a) WED with 1000 participants and DNA methylation data only, (b) WED with 1000 participants and genome only, (c) WED with 1000 participants and combined genome, microRNA expression and DNA methylation data (uniformly distributed), (d) WED with 10 positions and DNA methylation data only, (e) PC wit 10 positions and DNA methylation data only, and (f) ED with 1000 participants and combined biomedical data as in (c).

To further test our protocols' scalability, we evaluate their performance when performing a similar patient query against *1 million* patients in six different settings covering each of the six scenarios in Fig. 3. In the following, we report the setting and the corresponding total time to perform the similarity computation: (a) WED protocol for 100 positions on methylation data: 4.6 hrs, (b) WED protocol for 1,000 positions on DNA: 14.6 hrs, (c) WED protocol for 100 positions on the three data types: 2.3 hrs, (d) WED protocol for 10 positions on methylation data: 1.3 hr, (e) PC protocol for 10 positions on methylation data: 3.8 hrs, (f) ED protocol for 1,000 positions on the three data types: 10.8 hrs. These results demonstrate the ability of our protocols to scale to up to 1 million patients.

In order to show that our protocols can even be run on less powerful machines, we perform several experiments locally, i.e., on a MacBook Pro with 16 GB memory and 2,7 GHz Intel Core i7 processor. The same laptop is used for both the hospital and user. The resulting time is on average five times slower than with more powerful machines used in other experiments.

We also report the size of communication between the hospital and the user in Table 2. As expected, using different combinations of biomedical data or a different number of positions does not affect the size of data being exchanged because the size of ciphertexts is independent of the data type. The hospital or user always sends/responds with the same type and number of ciphertexts, independently of the data type used, only depending on which protocol is being executed. This is why we just report the data exchanged per patient. As the table shows, the PC protocol's communication size is larger compared to the other protocols. This is due to having two rounds of communication and having a ciphertext which is the result of squaring a ciphertext, compared to the other protocols that only have one round with standard ciphertexts being sent. The squared ciphertext has a larger size than a standard ciphertext due to its construction as described in Section 4.3. Finally, note that the data exchanged is slightly larger than expected due to the overhead added by Java.

We do not report the encryption time because the hospital encrypts the data only once per patient in its database. As already mentioned, this encryption occurs before any user can use the system and, as such, it is not time-critical. After the data is encrypted, users can start querying the hospital's encrypted data.

Finally, it is important to mention that we are considering the worst case for the number of queries. Indeed, according to our biomedical colleagues, medical practitioners or researchers are interested in having one to maximum three similar patients for a given case. On one hand, for common diseases or variants, there is a high chance that the user gets the one or those few patients he needs in 1,000 queries or before. On the other hand, there is some chance that she would have to query more than 1,000 patients before finding one or two matching patient(s) in the case of rare diseases. In order to account for the latter case, we have further experimented our similarity computation protocols with 1 million patients.

Second, the number of queries can be significantly decreased by clustering the hospital's data. The hospital can use clustering techniques to provide a few representative patients and let the user query against those only. If the SPQ results in a similar representative, a more fine-grained search in the cluster containing this representative can be carried out similarly.

However, the clustering technique mentioned in [35] would not work in our setting because it creates clusters based on the whole genome. Intuitively, our clustering idea is to run the clustering algorithm on the data multiple times, each time on different sets of positions. Each patient will have an ID to represent him, for clustering purpose only. The hospital would then publish the clusters' description, i.e., which positions are considered into this cluster, and attach a list next to each patient identifying which clusters he is in. The positions used for clusters can be for example the positions used for identifying different diseases, with every disease having its own clusters. Note that since the hospital knows its patients' values, the clustering would happen on plain data and only once, at the setup.

Table 2. Data exchanged per patient between the user and the hospital for the different protocol variants

Protocol	Data sent by user	Data sent by hospital
ED & WED	802 Bytes	47 Bytes
PC	3594 Bytes	849 Bytes

9 Countering Malicious Behavior

In this section, we propose possible defenses against malicious users and hospital.

9.1 Malicious Users

Our first defense aims to maintain the accuracy of the SPQs while enhancing the hospital's data protection. It is based on the fact that the users are not anonymous while querying the hospital. The hospital should create and store a profile for each user based on his expected behavior. Then, as the user is not anonymous while querying the hospital, the hospital monitors the user's actions in order to detect any deviation from proper behavior. If the hospital catches misbehavior, it can take actions ranging from blocking the user to legal actions.

Due to the strong privacy guarantees of our protocols, the hospital only learns which user is querying the service. However, it does not learn which positions are queried, which patient on the hospital's side is queried, or what the patient's data on the user side looks like. Depending on the protocol used, it does not learn either the exact similarity outcome. Therefore, it is challenging for the hospital to detect every kind of misbehavior. We will present here an extension for the defense mentioned above, which provides more capabilities to the hospital – effectively increasing the privacy of the hospital's data – by degrading the accuracy of the SPQs.

If we want to protect our scheme against a malicious user, we must assume he does not know the final similarity value. If the user knew the similarity outcome, it would be quite easy for him to query specific SNPs and appropriately assign his SNPs' values to infer those of the hospital. Therefore, in the following, we assume the setting in which only the hospital learns the similarity measure while the user only learns whether the similarity outcome is above or below a public threshold. In this setting, we can apply the *sparse vector technique* for differential privacy as defined in [51]. The sparse vector technique adds noise to the similarity values and compares the result to a noised threshold. Generally, it is assumed that queries resulting in a value above the threshold are considered privacy sensitive (also called a positive outcome). The noise on the threshold is only redrawn when the outcome is positive, and there is a bound c on the number of queries that yield such a positive outcome. It is straightforward to adapt the algorithm for the case where results below the threshold should be considered privacy-sensitive.

Given our similarity measures, we consider the case where query results above the threshold are considered privacy-sensitive. The sparse vector technique is (ϵ, δ)-differentially private and provides (α, β)-accuracy guarantees on the result. Let $a_1, \dots, a_k \in \{\top, \bot\}$ denote a stream of answers for k queries $x_1, \dots, x_k$, determining whether the similarity $s(x_i, h)$ between x_i and the hospital's values h is above the threshold ($\top$) or not ($\bot$). Given $\delta > 0$, the following equation defines the relation between (ϵ, δ)-differentially privacy and (α, β)-accuracy:

$$\alpha = \frac{(\ln k + \ln \frac{2c}{\beta} \sqrt{512 \ln \frac{1}{\delta}})}{\epsilon}.$$

The sparse vector technique is called (α, β)-accurate with respect to a threshold T if, except with probability at most β, the algorithm does not halt before the k^{th} query, and for all $a_i = \top$: $s(x_i, h) \geq T - \alpha$, and for all $a_i = \bot$: $s(x_i, h) \leq T + \alpha$. In practice, this means that the threshold has not been altered by more than α with probability $(1 - \beta)$.

There are six parameters influencing the algorithm: k, c, α, β, ϵ, δ. k is the total number of queries a user is potentially able to submit, and $c \leq k$ is the cut-off point on the number of privacy sensitive outcomes. If c queries returned a positive outcome (above the noised threshold), further queries will not be answered. Given this guarantee, the algorithm is able to answer these queries with a positive outcome up to an error α with probability $(1 - \beta)$. The parameter ϵ is the privacy-loss parameter, controlling the amount of noise. δ controls the probability of a privacy breach event happening and should be kept very small or event be set to 0 (the corresponding relation for $\delta = 0$ can be found in [51]).

By determining suitable bounds for the error on the threshold, the hospital can use the relation above to calculate suitable parameters for the sparse vector technique in order to increase the patient's data privacy while striving for an acceptable utility. How the parameters have to be chosen in practice mainly depends on the amount of error medical practitioners would feel comfortable with.

For example, a hospital could allow a user to perform a total of $k = 1,000$ queries, $c = 10$ of which may yield a positive outcome. From medical practice with a given type of data, the hospital determines that an error of at most $\alpha = 50$ is acceptable. The chosen error should only be allowed to exceed α in $\beta = 10\%$ of the cases. Setting $\delta = 10^{-10}$ will then yield $\epsilon \approx 11.64$.

9.2 Malicious Hospital

First, it is important to note that a malicious hospital can publish wrong data in the first place. However, as the hospital's encrypted data is published, such data cannot be changed at any time without anyone noticing it. This eliminates the risk of having the hospital publishing data specially crafted to steal a user's data.

For this defense, we consider the setting in which only the user learns the similarity measure, and the hospital learns nothing. The intuition behind the defense is forcing the hospital to perform all of its operations in public while presenting proof of correctness for them. The user can check that if the hospital is misbehaving by merely checking the operations and the proofs.

The hospital only decrypts the value in the ED scheme. To defend against a malicious hospital, the hospital would only need to provide a proof of decryption like the one presented in [52]. This way the user will be sure that the decryption is correct, then he can follow the scheme to unblind the value locally and learns the similarity measure, while the hospital learns nothing.

For the PC scheme, it is more complicated. In this scheme the hospital blinds p_{10} and p_{11} locally. To perform the blinding in public, the hospital would need to blind the values before decrypting them to avoid leaking their values to the user, and to provide a proof of correctness for the blinding. To accomplish this, the hospital can use a non-interactive zero-knowledge proof of knowledge (NIZK) [53]. More concretely, the hospital publishes $\mathcal{X}_1$ and $\mathcal{X}_2$ which correspond to the blinded versions of p_{10} and p_{11}, respectively. The hospital then proves the correctness of $\mathcal{X}_1$'s and $\mathcal{X}_2$'s construction by creating and publishing the following proof of knowledge $PK\{(r_h) : \mathcal{X}_1 = p_{10}^{r_h^2} \wedge \mathcal{X}_2 = p_{11}^{r_h}\}$. Concretely, this proof shows that there exists an r_h that the hospital knows and uses to construct $\mathcal{X}_1$ and $\mathcal{X}_2$ as illustrated in the proof. The hospital can decrypt both values in public while proving their decryption. Since the same blinding factor r_h is used in blinding both values, the user can calculate p_{12} himself, and the rest is executed as described in the original scheme.

10 Related Work

There are several previous papers on finding similar patients in a privacy-preserving manner.

Wang et al. [22] propose a technique for similar patient queries that uses the edit distance between DNA sequences approximated by a private set difference. Contrary to this approach, we are not bound to genomic data, but instead we consider other types of biomedical data such DNA methylation or microRNA expression data as well. These data types cannot be handled by Wang et al.'s approach due to its representation and the similarity measure they use. Furthermore, our approach improves upon the amount of communication needed and the flexibility to query specific positions selectively.

Al Aziz et al. [54] propose two approximations for edit distance to perform similar patient queries. Their work is similar to [22], with one of their approximations achieving a better accuracy for longer genomic sequences at the expense of the running time. Our protocol does not rely on the edit distance, which enables us to use other data types than DNA only. Also, it gives us the advantage to offer the user the possibility to apply different weights for different positions, which cannot be done using their approach.

Asharov et al. [23] present another SPQ improving upon the work in [22]. However, they also only consider DNA data. Moreover, their approach leaks a syntactic reference genome, which they claim does not leak any information but without any formal proof. Our approach does not leak any information about the result except that it is above or below a threshold, and we reduce the communication needed for the protocols. Besides, even if the syntactic reference genome does not leak information, it has another disadvantage: Each hospital will have a different reference genome, which forces the user to calculate and store the difference between each of his patients and all of these reference genomes.

Naveed et al. [24] give an interesting solution for a variety of operations like similar patient query, paternity test, etc. by relying on controlled functional encryption. Their approach presents a modified functional encryption which is efficient and can achieve the previously mentioned functions for DNA sequences. However, it would be hard to handle the methylation or expression data as they consider the patient as a vector of 1's and 0's which indicate the user's SNPs values. The challenge of methylation or expression data is that they are real, not binary, values. They further rely on a third party which does not collude with anyone, which is not required with our approach. Finally, we consider different settings regarding the leakage of the final result (as shown in Table 1).

Huang et al. [55] show that garbled circuits can be used for efficiently computing similar patient queries similarly to [22]. However, using homomorphic encryption makes our schemes more flexible than garbled circuits since garbled circuits require the number and set of positions to be determined in advance. Given the size of every data type, it is not practical to create a circuit for every possible combination of positions. Finally, contrary to our schemes, garbled circuits cannot prevent the leakage of the positions the user is interested in.

Finally, Oprisanu and De Cristofaro propose a privacy-enhanced version [56] of the Matchmaker Exchange (MME) platform [10]. More specifically, they rely on Reverse Private Information Retrieval (PIR) to allow researchers to query a gene within the MME platform anonymously. Contrary to their approach, we provide data confidentiality by hiding both the positions and values of the queried genomic variants.

Another field which handles similar problems as finding the similarity between two users is location privacy. In location privacy, the task is to find similar routes. Hallgren et al. present a scheme for privacy-preserving ridesharing based on threshold private set intersection (PSI) in [57]. Using PSI for comparing similar patients would have several drawbacks. First, it would restrict the similarity metrics used. Second, the user would need to reveal the set of positions he is interested in. Third, combining different data types with different weights would not be possible unless the user reveals the set of weights, which is hidden in our schemes.

Zhong et al. present three protocols for determining if two users are close to each other while preserving their location privacy [58]. They use homomorphic encryption similar to our approach. However, their use case is restricted to only finding the distance between two users. We are considering many positions and more complicated use cases. Moreover, their protocols suffer from disadvantages which would question the practicality if used in our use case, such as the need for a trusted third party or the cost on the results' accuracy.

11 Conclusion

In this paper, we develop the first cryptographic protocols able to perform privacy-preserving similar patient queries with new types of biomedical that will be used in daily routine by medical practitioners in the near future. Beyond genomic data, epigenomic and transcriptomic data reveal a lot about our personal health status and, as such, must be processed with the highest security and privacy guarantees. In this endeavor, we propose two schemes which rely on different similarity measures, namely the Euclidean distance and Pearson correlation coefficient, with the ability to query a specific subset of data and positions with different weights. Our schemes do not require any trusted third party and significantly reduce the online time needed by the hospital and user.

We prove the security of our schemes against honest-but-curious users and hospitals, and we implement and test our protocols to show their applicability in real settings based on real biomedical data. Our experimental results demonstrate that a typical number of positions of interest can be queried for 1,000 patients in less than five seconds for the weighted Euclidean distance and in less than thirty seconds for the Pearson correlation coefficient. Finally, we present strategies to mitigate the threat of malicious users or hospitals. We formalize our defense against malicious users by relying on the sparse vector technique which provides guarantees on the data privacy and on the results' accuracy.

12 Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ ERC grant agreement no. 610150-imPACT. This project was further supported by the grant #2017-201 of the Strategic Focus Area "Personalized Health and Related Technologies (PHRT)" of the ETH Domain.

References

[1] Shirley E. Poduslo, Rong Huang, Jie Huang, and Sierra M. Smith. Genome screen of late-onset alzheimer's extended pedigrees identifies trpc4ap by haplotype analysis. *American Journal of Medical Genetics Part B: Neuropsychiatric Genetics*, 150B(1):50–55, 2009.

[2] Andrew P Feinberg and M Daniele Fallin. Epigenetics at the crossroads of genes and the environment. *JAMA*, 314:1129–1130, 2015.

[3] Peter A Jones and Stephen B Baylin. The epigenomics of cancer. *Cell*, 128:683–692, 2007.

[4] Irfan A Qureshi and Mark F Mehler. Advances in epigenetics and epigenomics for neurodegenerative diseases. *Current neurology and neuroscience reports*, 11:464–473, 2011.

[5] Manel Esteller and James G. Herman. Cancer as an epigenetic disease: Dna methylation and chromatin alterations in human tumours. *The Journal of Pathology*, 196(1):1–7, 2002.

[6] Jun Lu, Gad Getz, Eric A Miska, Ezequiel Alvarez-Saavedra, Justin Lamb, David Peck, Alejandro Sweet-Cordero, Benjamin L Ebert, Raymond H Mak, Adolfo A Ferrando, et al. Microrna expression profiles classify human cancers. *nature*, 435(7043):834–838, 2005.

[7] Mohamed Hamed, Christian Spaniol, Alexander Zapp, and Volkhard Helms. Integrative network-based approach identifies key genetic elements in breast invasive carcinoma. *BMC Genomics*, 16(5), 2015.

[8] Nora K. Speicher and Nico Pfeifer. Towards multiple kernel principal component analysis for integrative analysis of tumor samples. *ArXiv e-prints*, January 2017.

[9] Nora K. Speicher and Nico Pfeifer. Integrating different data types by regularized unsupervised multiple kernel learning with application to cancer subtype discovery. *Bioinformatics*, 31(12):i268, 2015.

[10] Anthony A Philippakis, Danielle R Azzariti, Sergi Beltran, Anthony J Brookes, Catherine A Brownstein, Michael Brudno, Han G Brunner, Orion J Buske, Knox Carey, Cassie Doll, et al. The matchmaker exchange: a platform for rare disease gene discovery. *Human mutation*, 36(10):915–921, 2015.

[11] Zhen Lin, Art B Owen, and Russ B Altman. Genomic research and human subject privacy. *Science*, pages 183–183, 2004.

[12] Erman Ayday, Emiliano De Cristofaro, Jean-Pierre Hubaux, and Gene Tsudik. Whole genome sequencing: Revolutionary medicine or privacy nightmare? *Computer*, pages 58–66, 2015.

[13] Muhammad Naveed, Erman Ayday, Ellen W Clayton, Jacques Fellay, Carl A Gunter, Jean-Pierre Hubaux, Bradley A Malin, and XiaoFeng Wang. Privacy in the genomic era. *ACM Computing Surveys (CSUR)*, 48:6, 2015.

[14] Yaniv Erlich and Arvind Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15:409–421, 2014.

[15] Mathias Humbert, Kévin Huguenin, Joachim Hugonot, Erman Ayday, and Jean-Pierre Hubaux. De-anonymizing genomic databases using phenotypic traits. *Proceedings on Privacy Enhancing Technologies*, 2015(2):99–114, 2015.

[16] Michael Backes, Pascal Berrang, Mathias Humbert, Xiaoyu Shen, and Verena Wolf. Simulating the large-scale erosion of genomic privacy over time. *IEEE/ACM transactions on computational biology and bioinformatics*, 2018.

[17] Eric E Schadt, Sangsoon Woo, and Ke Hao. Bayesian method to predict individual SNP genotypes from gene expression data. *Nature genetics*, 44:603–608, 2012.

[18] Michael Backes, Pascal Berrang, Anne Hecksteden, Mathias Humbert, Andreas Keller, and Tim Meyer. Privacy in epigenetics: Temporal linkability of MicroRNA expression profiles. In *Proceedings of the 25th USENIX Security Symposium*, 2016.

[19] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in MicroRNA-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 319–330. ACM, 2016.

[20] Michael Backes, Pascal Berrang, Matthias Bieg, Roland Eils, Carl Herrmann, Mathias Humbert, and Irina Lehmann. Identifying personal dna methylation profiles by genotype inference. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 957–976. IEEE, 2017.

[21] Pascal Berrang, Mathias Humbert, Yang Zhang, Irina Lehmann, Roland Eils, and Michael Backes. Dissecting privacy risks in biomedical data. In *Proceedings of the 3rd IEEE European Symposium on Security and Privacy (Euro S&P). IEEE*, 2018.

[22] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 492–503, New York, NY, USA, 2015. ACM.

[23] Gilad Asharov, Shai Halevi, Yehuda Lindell, and Tal Rabin. Privacy-preserving search of similar patients in genomic data. Cryptology ePrint Archive, Report 2017/144, 2017. http://eprint.iacr.org/2017/144.

[24] Muhammad Naveed, Shashank Agrawal, Manoj Prabhakaran, XiaoFeng Wang, Erman Ayday, Jean-Pierre Hubaux, and Carl Gunter. Controlled functional encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1280–1291, New York, NY, USA, 2014. ACM.

[25] Yadong Yang, Edward Ruiz-Narvaez, Peter Kraft, and Hannia Campos. Effect of apolipoprotein e genotype and saturated fat intake on plasma lipids and myocardial infarction in the central valley of costa rica. *Human Biology*, 79(6):637–647, 2017/06/23 2007.

[26] María J Artiga, María J Bullido, Isabel Sastre, María Recuero, Miguel A García, Jesús Aldudo, Jesús Vázquez, and Fernando Valdivieso. Allelic polymorphisms in the transcriptional regulatory region of apolipoprotein e gene. *FEBS Letters*, 421(2):105–108, 1998.

[27] Gerwin Roks, Marc Cruts, Jeanine J. Houwing-Duistermaat, Bart Dermaut, Sally Serneels, Louis M. Havekes, Albert Hofman, Monique M. B. Breteler, Christine Van Broeckhoven, and Cornelia M van Duijn. Effect of the apoe-491a/t promoter polymorphism on apolipoprotein e levels and risk of alzheimer disease: The rotterdam study. *American Journal of Medical Genetics*, 114(5):570–573, 2002.

[28] Simon M. Laws, Eugene Hone, Sam Gandy, and Ralph N. Martins. Expanding the association between the apoe gene and the risk of alzheimer's disease: possible roles for apoe promoter polymorphisms and alterations in apoe transcription. *Journal of Neurochemistry*, 84(6):1215–1236, 2003.

[29] June E. Eichner, S. Terence Dunn, Ghazala Perveen, David M. Thompson, Kenneth E. Stewart, and Berrit C. Stroehla. Apolipoprotein e polymorphism and cardiovascular disease: A huge review. *American Journal of Epidemiology*, 155(6):487, 2002.

[30] Anna Danielsson, Szilárd Nemes, Magnus Tisell, Birgitta Lannering, Claes Nordborg, Magnus Sabel, and Helena Carén. Methped: a dna methylation classifier tool for the identification of pediatric brain tumor subtypes. *Clinical Epigenetics*, 7(1):62, 2015.

[31] Dario Catalano and Dario Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1518–1529, New York, NY, USA, 2015. ACM.

[32] Ancestry. https://www.ancestry.com/dna/. Accessed: 2017-07-25.

[33] 23andme. https://www.23andme.com/en-int/ancestry/. Accessed: 2017-07-25.

[34] John Quackenbush. Computational genetics: computational analysis of microarray data. *Nature reviews genetics*, 2(6):418, 2001.

[35] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333–337, 2014.

[36] Burkhard Morgenstern, Bingyao Zhu, Sebastian Horwege, and Chris André Leimeister. Estimating evolutionary distances between genomic sequences from spaced-word matches. *Algorithms for Molecular Biology*, 10(1):5, Feb 2015.

[37] Jianchao Yao, Chunqi Chang, Mari L. Salmi, Yeung Sam Hung, Ann Loraine, and Stanley J. Roux. Genome-scale cluster analysis of replicated microarrays using shrinkage correlation coefficient. *BMC Bioinformatics*, 9(1):288, Jun 2008.

[38] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[39] dbSNP. https://www.ncbi.nlm.nih.gov/SNP/.

[40] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *22nd Network and Distributed System Security Symposium (NDSS' 15)*, 2015.

[41] Paul J McLaren, Jean Louis Raisaro, Manel Aouri, Margalida Rotger, Erman Ayday, István Bartha, Maria B Delgado, Yannick Vallet, Huldrych F Günthard, Matthias Cavassini, et al. Privacy-preserving genomic testing in the clinic: a model using HIV treatment. *Genetics in Medicine*, 2016.

[42] George Danezis and Emiliano De Cristofaro. Fast and private genomic testing for disease susceptibility. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 31–34. ACM, 2014.

[43] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[44] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag.

[45] Florian Kerschbaum and Orestis Terzidis. Filtering for private collaborative benchmarking. In Günter Müller, editor, *Emerging Trends in Information and Communication Security*, pages 409–422, Berlin, Heidelberg, 2006. Springer

Berlin Heidelberg.
[46] Personal genomes project (PGP) platform. https://my.pgp-hms.org.
[47] Gene expression omnibus (GEO). https://www.ncbi.nlm.nih.gov/geo/.
[48] Sally R. Lambert, Hendrik Witt, Volker Hovestadt, Manuela Zucknick, Marcel Kool, Danita M. Pearson, Andrey Korshunov, Marina Ryzhova, Koichi Ichimura, Nada Jabado, Adam M. Fontebasso, Peter Lichter, Stefan M. Pfister, V. Peter Collins, and David T. W. Jones. Differential expression and methylation of brain developmental genes define location-specific subsets of pilocytic astrocytoma. *Acta Neuropathologica*, 126(2):291–301, Aug 2013.
[49] Petra Leidinger, Valentina Galata, Christina Backes, Cord Stähler, Stefanie Rheinheimer, Hanno Huwer, Eckart Meese, and Andreas Keller. Longitudinal study on circulating mirnas in patients after lung cancer resection. In *Oncotarget*, 2015.
[50] Christine Jost, Ha Lam, Alexander Maximov, and Ben J. M. Smeets. Encryption performance improvements of the paillier cryptosystem. *IACR Cryptology ePrint Archive*, 2015:864, 2015.
[51] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.
[52] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.
[53] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.
[54] Md Momin Al Aziz, Dima Alhadidi, and Noman Mohammed. Secure approximation of edit distance on genomic data. *BMC Medical Genomics*, 10(2):41, Jul 2017.
[55] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*. The Internet Society, 2012.
[56] Bristena Oprisanu and Emilliano De Cristofaro. Anonimme: Bringing anonymity to the matchmaker exchange platform for rare disease gene discovery. *bioRxiv*, 2018.
[57] Per Hallgren, Claudio Orlandi, and Andrei Sabelfeld. Privatepool: Privacy-preserving ridesharing. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 276–291, Aug 2017.
[58] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In Nikita Borisov and Philippe Golle, editors, *Privacy Enhancing Technologies*, pages 62–76, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

A Reversed SPQ

Reversing the roles of hospital and user results in an additional protocol, which is useful by itself. Suppose that a physician wants to test a patient against some disease. The other party (a hospital or a third party laboratory) has a private model (i.e., positions and similarity measure) they do not wish to reveal. In such a scenario, the physician could act in the role of our protocol's hospital and reveal his encrypted data to the other party. The other party then utilizes the protocol (as described in Section 6.2 and 6.3) to test the patient's data for a specific disease. In the end, the physician is the one decrypting the final result and learning the diagnosis.

B Security Analysis

In this section, we analyze and prove the security of our building blocks and protocols. We define security as follows: the user should not learn anything about the hospital's data except the final outcome of the schemes. Similarly, the hospital should not learn anything about the user's data except the final outcome of the scheme. We formalize security as the following cryptographic game between a challenger and an adversary:

Definition 1 (Security). *A similar patient query π is secure if and only if, for all polynomial-time (PPT) adversaries* A, *there is a negligible function* negl *such that:*

$$\left|\Pr[\mathsf{ExpSec}_{\pi}^{\mathsf{A}}(\kappa, \mathsf{b}) = 1]\right| \leq 0.5 + \mathsf{negl}(\kappa)$$

Where κ is the security parameter, π is the similarity metric, and $\mathsf{ExpSec}_{\pi}^{A}(\kappa, b)$ denotes the following experiment:
Setup: *The challenger sets up the system as defined in Section 6.1.*
Queries: *The challenger provides* A *with the following two interfaces:*
– *On input* $\mathsf{CorruptUser}$, *the challenger aborts if* $\mathsf{CorruptHospital}$ *interface was called before, otherwise it passes the control over the user to* A *and plays the role of the hospital in the specified protocol as defined in the scheme π.*
– *On input* $\mathsf{CorruptHospital}$, *the challenger aborts if* $\mathsf{CorruptUser}$ *interface was called before, otherwise it passes the control over the hospital to* A *and plays the role of the hospital in the specified protocol as defined in the scheme π.*
Corrupting a party in the game means that A *will be able to see any messages that this party can see. However, the adversary still has to follow the protocol correctly due to the honest-but-curious setup.*

Challenge: *At some point,* A *wants to be challenged on two patients (i and j), the data of which have the same dimensions and type. If i and j have different data dimensions, the challenger aborts. Otherwise, it flips a coin* b *and picks a user to encrypt his data and publish the encryption.* A *has to follow the protocol in her role, but can try to learn extra information from any value received or generated by the party she has corrupted. The protocol is run normally till the point the similarity measurement is learned. We do not allow* A *to learn the result because by learning the result she can trivially win the game by just running the scheme against the user i and j and checking the output. Then,* A *outputs the bit* b' *which represents which patient he thinks the challenger picked, challenger outputs* 1 *if and only if* $\mathsf{b} = \mathsf{b}'$*.*

Before proving our schemes to be secure, we next recall the definition of CPA security.

B.1 CPA security

The CPA property stands for indistinguishability of ciphertexts under chosen plaintext attack. This property is a basic property for any secure public key cryptosystem. Intuitively, this property means that the scheme is secure even if an attacker has an encryption oracle. It is considered basic in public key cryptosystems, because the public key is by definition public, effectively corresponding to an encryption oracle.

A public key encryption scheme π is called CPA secure if, for all polynomial-time (PPT) adversaries A, there is a negligible function negl such that:

$$\left| \Pr[\mathsf{ExpCPA}^A_\pi(\kappa, b) = 1] \right| \leq 0.5 + \mathsf{negl}(\kappa)$$

Where κ is the security parameter, and $\mathsf{ExpCPA}^A_\pi(\kappa, b)$ denotes the following experiment:

Experiment 1 ($\mathsf{ExpCPA}^A_\pi(\kappa, b)$).

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_\pi(1^\kappa)$$
$$(m_0, m_1) \leftarrow A(\mathsf{pk})$$
$$c_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$$
$$b' \leftarrow A(c_b)$$

Output 1 if and only if $|m_0| = |m_1| \wedge b' = b$*.*

B.2 Squaring modification proof

Having recalled the CPA property, we will first show our squaring modification to yield a CPA secure scheme.

Proof of Theorem 2. Intuitively, if we are able to replace the output of the extension with the encryption of uniformly random numbers and the adversary cannot distinguish the difference, then the modification is secure.

The output of the extension consists of two ciphertexts β and α. In order to prove that we can replace β and α with the result of squaring a uniformly random number, we design a cryptographic game. In this game, the adversary is given the public key. Then, the adversary gives a number he wants to be squared. The game flips a coin b and sends him β' and α' which are the result of squaring the encryption of a uniformly random number if $b = 1$, or the result β and α of squaring his encrypted, chosen number if $b = 0$. The adversary outputs b' and he wins only if b' is equal to b.

We assume towards contradiction that there is a distinguisher D which can differentiate between the two cases with a non-negligible probability. We build a reduction R against the CPA property of Π_{PKE}.

Simulation

R initializes the CPA challenger to get the public key pk, which it forwards to D and waits until D sends a number m. R generates a uniformly random r and forwards m, r as its challenge and waits to receive c^*. R squares the ciphertext c^* as mentioned in Section 4.3 to get α^* and β^*. It forwards them to D and waits for the output of D. Whenever D outputs b, it simply forwards it as its own output.

Analysis

R is efficient and perfectly simulates both cases of D. In the first case, when $b = 0$, A receives β and α, which are the result of squaring m. In the second case, when $b = 1$, A receives β^* and α^*, which are the result of squaring a uniformly random number.

Since D wins with a non-negligible probability, R also wins with a non-negligible probability. This is a contradiction to the CPA property of Π_{PKE}. Therefore, the modification is secure.

□

B.3 SPQ using Euclidean distance

Proof of Theorem 3. To prove the security of this protocol we will consider the two possible views of the adversary. The first when he corrupts the hospital and the second when he corrupts the the user, which is simulated in the game by the adversary calling the CorruptHospital or CorruptUser interface. Since the game only allows a single entity to be corrupted every run, we will prove every case separately and show that both cases the adversary cannot win with a probability more than $0.5 + \mathsf{negl}(\kappa)$.

Let the two events CorruptHospitalWin and CorruptUserWin represent the events of winning the game when the adversary called the CorruptHospital or CorruptUser interface, respectively.

First we start with the simulation of the game, then we calculate the winning probabilities for each event. Assume towards contradiction the existence of an attacker A, who can break the SPQ using Euclidean distance with a probability greater than $0.5+\mathsf{negl}(\kappa)$, where negl is a negligible function. We build a reduction R which is an attacker against the CPA property of Π_{PKE}.

Simulation
R initializes the CPA challenger to get the public key pk, which it publishes. Since this is a public encryption scheme, A can encrypt all the messages she wants and does not need an encryption oracle. R waits for A to send the two patient data P_0, P_1 and aborts if these are not of the same dimension. Otherwise, R parses them to two sets of messages m_0, m_1, where the m_i is the biomedical data for P_i. R then forwards m_0, m_1 as its challenge messages set and receives c_b, which it publishes online. It is important to maintain the order of the messages because each item of this set would be the biomedical data of a certain position and type. R waits for A to corrupt a party using the interface provided, then according to which party was corrupted, R takes the role of the uncorrupted party and interacts with A as specified by the protocol except that it does not send the final outputs. R waits for A to output b' which represents A's guess on which patient it is. R then forwards the b' as its own output which represents which message is encrypted in c_b.

Analysis
We start with the CorruptHospitalWin event. Here A corrupts the hospital and R simulates the user. Since we do not consider the final value, A's view is just the encryption of the similarity value. The simulation here is simple since R just calculates the Euclidean distance between the encrypted values as indicated in Section 6.2 and sends A the final encrypted value.

In both cases $b = 0$ and $b = 1$, R perfectly simulates the game for A. The order of messages are preserved because the CPA game does not change the order of messages. Since A wins with a probability greater than $0.5 + \mathsf{negl}(\kappa)$ then R wins with the same probability. As R is efficient and wins the CPA game with a probability greater than $0.5+\mathsf{negl}(\kappa)$, this contradicts the CPA property of Π_{PKE}. Therefore, $P[CorruptHospitalWin] < 0.5 + \mathsf{negl}(\kappa)$.

For the CorruptUserWin event, A corrupts the user and R simulates the hospital. Since we do not consider the final value so A's view ends after sending the encryption of the similarity value. A just calculates the Euclidean distance between the encrypted values as indicated in Section 6.2 and sends R the final encrypted value.

The probability analysis here is the same as for the CorruptHospitalWin event. R simulates the game perfectly so the winning probability of A is the same as R, which contradicts the CPA property of Π_{PKE}. Therefore, $P[CorruptUserWin] < 0.5 + \mathsf{negl}(\kappa)$.

Finally, since both events CorruptHospitalWin and CorruptUserWin winning probability is less than $0.5 + \mathsf{negl}(\kappa)$, the SPQ using Euclidean distance protocol is secure.

□

B.4 SPQ using Pearson's coefficient

Proof of Theorem 4. Abstractly, this variant is similar to the other one which uses the Euclidean distance as a measurement of similarity. The only differences are that the parties perform another set of operations on the published ciphertext. But this is done locally, so it does not make a difference in the proof. There is a round of interaction in this protocol, which we will give a proof for. We consider the same events as mentioned in Section B.3.

We first consider the CorruptHospitalWin event. This case is more complex than the other variant as the hospital receives additional data from the user, which the adversary will learn. The hospital receives two ciphertexts which it can decrypt to get the two values of the nominator and denominator of the Pearson's coefficient. However, these two values are blinded by two randoms with at least the same length. This perfectly hides the values, since new randoms are picked for every value sent by the user. It perfectly hides since one can find $r_1 \cdot r_2 = r_1' \cdot userData$, assuming that the $userData$ is the data the user sent to the hospital and r_1, r_2 and r_1'

are random numbers. Therefore, an adversary corrupting the hospital does not learn anything about the user except the final result.

For the other event CorruptUserWin. A similar information-theoretic hiding argument can be applied on the p_{13} he receives because the hospital blinds it with a random number. This only leaves the part of the protocol before sending p_{12} which does not include any interaction between the user and hospital. Thus, the rest of the proof can be carried out analogously to the corresponding part in the SPQ with Euclidean distance's proof. Hence, the SPQ using Pearson's coefficient protocol is secure.

$\square$