Outsourcing Medical Dataset Analysis: A Possible Solution

Gabriel Kaptchuk(✉), Matthew Green, and Aviel Rubin

John Hopkins University, Baltimore, USA
{gkaptchuk,mgreen,rubin}@cs.jhu.edu

Abstract. We explore the possible ways modern cryptographic methods can be applied to the field of medical data analysis. Current systems require large computational facilities owned by the data owners or excessive trust given to the researchers. We implement one possible solution in which researchers operate directly on homomorphically encrypted data and the data owner decrypts the results. We test our implementation on large datasets and show that it is sufficiently practical that it could be a helpful tool for modern researchers. We also perform a heuristic analysis of the security of our system.

1 Introduction

Modern medical dataset analysis methods take large sets of medical records and attempt to extract truths about the underlying population. Because of the sensitive nature of the data being analysed and regulations requiring strict limitations on the sharing of that data, it is difficult for researchers to share datasets. Today, it can take up to a *year* before a researcher can actually begin the computational process of analyzing a dataset that they did not collect on their own. Data is often shared in sanitized form, with much of the data removed; this sanitization process requires time, labor and statistical expertise. Some data owners have chosen to allow researchers to send their queries to the data owners, who perform the analysis on the researcher's behalf. The process of analyzing medical datasets requires large amounts of computation on the part of the data owner for each question posed by a researcher. To best serve the medical research community, data owners must acquire technical expertise to properly anonymize and maintain datasets or contract a trusted third party to do it for them.

In this work we consider an institutional medical researcher, such as a member of a university or the research division of a company, interested in answering some query but who is without access to the required data. While it may be infeasible to independently gather data, it is likely that there exists a dataset containing sufficient information to answer the researcher's query. The data owner may want to share with the researcher but because the information is related to the medical history of patients, and therefore considered sensitive, sharing that dataset may be a complicated process.

We explore existing cryptographic methods in an effort to tackle the two main problems with the current way of sharing medical data. First, we wish to

A. Kiayias (Ed.): FC 2017, LNCS 10322, pp. 98–123, 2017.
https://doi.org/10.1007/978-3-319-70972-7_6

move the burden of cost from data owners to the researchers who want access to the data. All modern solutions that properly secure patient data require data owners to make large investments in hardware and technical expertise. While it is possible for a data owner to recoup those costs over time, requiring large startup costs deters the sharing of data and charging for access to the dataset limits the kinds of researchers able to use it. Second, it takes far too long for a researcher to acquire and analyze a dataset that has been properly anonymized and certified. Even after proper permission has been acquired, it may be extremely inconvenient to actually run analysis or to tweak the nature of the researcher's query.

2 Objectives

In order to build something useful to the medical research community, we attempt evaluate the usefulness of Fully Homomorphic Encryption while still ensuring the following six properties. These objectives were derived from conversations with professionals working in the medical research industry. Additionally, the analysis we ran to confirm that our system was practical enough to be used by members of the medical research community were also informed by these conversations.

Authenticity of results - the results obtained by the researcher should be as authentic and accurate as possible without compromising the privacy of individuals.
A rich range of possible analyses - virtually any analytical technique should be possible to the researcher. More formally, the limits on the possible set of operations should depend only on the parameters chosen for the FHE scheme.
Minimal computation on the part of the data owner - the computational responsibility of the data owner should be almost entirely limited to preprocessing the dataset a single time. We propose that a data owner should only have to provide a single server to allow for large numbers of researchers to perform analysis.
Privacy for individuals in the dataset - it should be impossible for a researcher with auxiliary information to learn anything about an individual in the population using legitimate analysis techniques. Specifically, we invoke differential privacy to protect information about individuals.
Security against adversarial researchers - an adversarial researcher attempting to extract information about individuals from the dataset should be caught with very high probability.
Practicality - our system should shorten the time it takes for a researcher to conceive of a researcher question to when their computational analysis has finished. The actual time it takes for a single run of the analysis process may take longer than current methods, providing this overall time shrinks.

While many existing solutions address some subset of these objectives, none accomplish all of them. In particular, existing systems lack practicality, proper cost distribution or a large space of possible computation. Anonymization presents security concerns and lacks practicality due to the long wait times for dataset acquisition. Analysis as a service requires a misappropriation of costs between the researcher and the data owner. Attempts like [19] have managed to be both practical and to outsource computation, but failed to allow for rich space of analytical techniques required by the medical industry. Our construction satisfies all the requirements of researchers in the medical industry.

3 Background

To understand our motivation, it is important to consider the ways in which modern medical dataset analysis is done. The reality of current analysis systems is that they are both extremely expensive for the data owner and take a long time for the query of a researcher to be fully answered. Researchers interested in fields as diverse as drug side effects, public health and genetics all utilize the large amounts of data regularly collected by medical professionals, insurance companies, or governments to make new discoveries or confirm theories. Under ideal circumstances, analysis is done with large sample sizes - discussions with professionals in the field lead us to believe that most studies utilize around 100,000 data points. The analytical models used by researchers vary from simplistic count and average to complex regressions or machine learning. While complex methods are gaining in popularity, measurements like regression, covariance and averages remain the primary tools employed by researchers.

There are various practical constructions employed to allow external researchers access to private data. The obvious, simple, and clearly insecure solution is to naively share the data without any security. While efficient, this makes the assumption that the researcher is a trusted party, which is often invalid.

3.1 Anonymization

Anonymization is a technique in which large amounts of information, hopefully all irrelevant, is purged before a dataset is shared with a researcher. The goal of anonymization is to allow an untrusted third party to confirm results or perform original analysis without revealing any personally identifiable information. The process is computationally expensive because it requires a data owner to reprocess the dataset each time a researcher posits a new query. For example, a researcher may start the process interested in a certain subset of the information about each patient only to later decided that other qualities of each patient are also required to confirm their hypothesis. This method also makes it extremely expensive for a researcher to explore a dataset without a specific hypothesis in mind. Additionally, there have been recent results showing that anonymization is not as secure as previously thought [28]. While a single instance of an

anonymized dataset leaks minimal information under most circumstances, combining it with a version of the same dataset anonymized for a different query can certainly allow a malicious researcher to compromise the privacy of individuals.

3.2 Analysis as a Service

This model has becoming increasingly popular recently as the medical community has adopted cloud technologies. Data owners or trusted third parties provide a service through which researchers are able to submit requests for work. The data owners or their surrogates then perform the computation over the dataset stored as plaintext. This requires data owners to acquire the technical expertise to maintain the system. More importantly, this forces data owners to shoulder the cost of providing their data to the medical research community or possibly charge researchers for the use of their data which would discourage collaboration.

3.3 Cost Consideration

While both anonymization and analysis as a service are common models for sharing statistical datasets, cutting-edge systems combine both techniques. The largest data owners maintain massive datasets on expensive computational infrastructure. When a researcher wants to answer some new query, they access the infrastructure itself, either physically or over a secure channel. Then, based on the requirements of their query, they select a certain anonymization of the dataset to use. A certain anonymization of the data may leave more information about income, but may contain little geographical information. Each time a new anonymization of the data is required by a researcher, the data owners must prepare a new subset of the data and get statisticians to certify it.

Once an appropriate version of the dataset has been prepared, the analysis is run on the data owner's systems. Because of inference attacks, allowing researchers to remove even anonymized datasets can be dangerous, especially when the researcher is likely to return to the same data owner to perform a different analysis soon afterwards. The two main concerns addressed in this work are time and cost. It is not uncommon for the time between the conception of a question and the moment when computational analysis begins to be months or even a year.

It is nearly inevitable that research will involve high costs for at least some of the parties involved. While typically one might assume that the burden of cost should be on the researchers themselves, given that they are the ones directly benefiting from computation, it is often the data owners who are forced to acquire expertise and infrastructure to service the researcher community. One company with which we spoke had $1 million in hardware costs to support the needs of researchers. While costs might eventually be recouped by charging researchers for use of the dataset, the costs from purchasing hardware alone may make it infeasible for a data owner to securely share their data. Especially if their dataset becomes desirable to many researchers, the costs of scaling up their operations quickly make it impossible to support widespread interest.

3.4 Existing Cryptographic Options

In order to construct a system that addresses the problems above, we call upon existing cryptographic primitives and systems. Some, like differential privacy, have been widely used in the field and their limitations are well understood. The practicality of others, like FHE and homomorphic signatures, has yet to be fully tested. Because we are attempting to build a practical system that minimizes the amount of time between the medical researcher's initial query and receiving the final answer, we choose our cryptographic primitives carefully. Additionally, various primitives may be helpful in achieving some of the objectives in Sect. 2 but may prohibit the achievement of others. We give a broad summary of the cryptographic methods chose to use in our case study below and include methods we chose not to utilize in Appendix D.

3.5 Fully Homomorphic Encryption

FHE allows for addition and multiplication on encrypted data without the need for decryption. The first construction of FHE was published in [21] but was too inefficient for practical computation. Subsequent efforts, most notably the BGV construction in [9], have attempted to increase the efficiency and modern constructions are teetering on the edge of practicality. To make the schemes more usable, there has been a push towards "leveled" homomorphic encryption schemes which can compute a certain depth of circuit before inherent noise renders the ciphertext useless. For a full background on the intricacies of FHE and a more complete list of citations, refer to [33].

Smart and Vercauteren proposed an addition to the BGV FHE in [31], in which many plaintext values could be encoded into a single ciphertext. To do this, the plaintext values are put into a vector and all additions and multiplications are computed entrywise. This allows for single instruction multiple data operations and significantly increasing the efficiency of the scheme. Our implementation requires that the FHE scheme used supports Smart-Vercauteren plaintext packing and for the rest of this work all homomorphic operations can be considered to be done within this framework.

3.6 HELib

The best available implementation of a modern leveled FHE is the C++ library HELib. While most of the code currently written using HELib implements relatively simple computations, our testing shows that is both robust and reasonably efficient for more complex computations. The FHE scheme it implements encodes integers into BGV ciphertext, supporting addition and multiplication operations. The underlying plaintext values are added and multiplied modulo some prime. The choice of primes, the maximum level of the circuit, and security parameter all influence the size of the ciphertext and the efficiency of the operations. Details about the use of HELib and the FHE scheme it implements can be found at [23].

3.7 Differential Privacy

Differential privacy prevents an attacker from learning anything about individuals while still gleaning meaningful information from aggregated statistics. This is not the only notion of privacy that can be applied to statistical datasets, but it has recently become the most popular. With the rise of laws requiring the protection medical data, ensuring it is impossible to recover the information of any given individual effectively shields data owners from legal action. We give a more detailed background of differential privacy in Appendix C.

4 Construction

We assume a data owner $\mathcal{D}$ with a medical dataset D_{initial} of vectors $\boldsymbol{d} \in \mathbb{R}^n$. $\mathcal{D}$ transforms the dataset into the proper format, encrypts it using fully homomorphic encryption as $D^* = \mathsf{Encrypt}(\mathsf{D}_{\text{formatted}})$ and publishes it on the internet. A researcher $\mathcal{R}$ then prepares a program to be run on the dataset, described in the form of a transcript T and performs the computation $T(D^*)$. The result of this computation is a ciphertext c with an embedded integrity check and transmits c to $\mathcal{D}$. Finally $\mathcal{D}$ verifies that T and c match, computes the decryption, adds noise to guarantee differential privacy and sends this final result to $\mathcal{R}$. A protocol diagram can be found in Appendix B.

4.1 Dataset Formatting

We assume that the data owner $\mathcal{D}$ has some set of $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_{|D|}\}$ s.t. $\boldsymbol{d}_i \in \mathbb{R}^n$ where each dimension of $\boldsymbol{d}_i$ represents part of the medical record for patient i. Each vector $\boldsymbol{d}$ is made up of data entries α and binary entries β. The data entries are real valued and represent information about the patient like age, blood pressure or income. The binary entries represent the qualities of a patient, like the presence of a medication or medical condition.

$$D_{\text{initial}} = \left\{ \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_m \\ \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_{n-m} \end{pmatrix} \cdots \begin{pmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_m \\ \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_{n-m} \end{pmatrix} \right\}$$

If $\mathcal{D}$ has a dataset that is formatted differently, it is clear how to transform any dataset into this format. The only intricacy of this transformation is that all values in the vector must be integer valued, while real medical datasets also contain both text and real-number values. For the most part, this problem can be easily solved while only losing a little granularity in the data. Real-number values can be scaled and rounded such that the information is still rich enough to convey meaning. Text data can either be automatically binned, adding a β value for each possible value of that text field, or can be manually translated into a integer scale as appropriate.

4.2 Data Binning

Data binning beings with $\mathcal{D}$ dividing the range of each data entry α_i into continuous disjoint bins $\{\beta_1^{\alpha_i}, \beta_2^{\alpha_i}, \ldots, \beta_{b_i}^{\alpha_i}\}$, where the number of bins b_i is chosen separately for each α_i. $\mathcal{D}$ then inserts a new row into the data set for each $\beta_j^{\alpha_i}$ and sets the $\beta_j^{\alpha_i} = 1$ containing the value for α_i for each α_i. For example, if α_i represents age, $\mathcal{D}$ might create $\beta_j^{\alpha_i}$ as 5 year bins from 0 to 100. A patient of age 37 would have $\beta_8^{\alpha_i} = 1$ and $\beta_j^{\alpha_i} = 0\ \forall j \neq 8$.

The increased number of bins for each α give researchers greater granularity of possible computations but also increases the size of the dataset. Because this dataset will be prepared only once, the data owner chooses the maximum possible granularity for all researchers at once. Many fields, like age or income, have natural bin sizes while other fields will be completely at the discretion of the data owner.

4.3 Integrity Check Embedding and Encryption

The FHE scheme used in encrypting the dataset should include Smart-Vercauteren plaintext packing. This property allows a vector of plaintext values to be encrypted into a single ciphertext and all operations are computed entry-wise. The length l of the plaintext vectors is determined by the various parameters to the FHE scheme, but in general we will consider vectors of about 1000 values.

Each plaintext vector contains values from a single row of the database (i.e. a specific α or β from multiple patients). Each vector begins $\frac{l}{2}$ values from the dataset, in the order listed in the dataset. Thus, the first ciphertext will be an encryption of the α_1 entry from the first $\frac{l}{2}$ patient record; the second will be the α_1 entries from the next $\frac{l}{2}$ patient records, and so on. For each such vector, $\mathcal{D}$ embeds the tools to allow for rapid verification. $\mathcal{D}$ selects a random value π and a random permutation Φ, both to be used for all vectors in the D. For each entry e in the vector $\boldsymbol{v}$, $\mathcal{D}$ computes $e' = \pi e \mod p$, where p is a prime and a parameter to the FHE scheme, and appends that value to $\boldsymbol{v}$. Next, $\mathcal{D}$ appends a different random value k to the end of each vector and records k for each vector. Finally, $\mathcal{D}$ applies Φ it to all vectors in D.

$$\Phi\left(\alpha_1^1\ \pi\alpha_1^1\ \alpha_1^2\ \pi\alpha_1^2\ \alpha_1^2\ \pi\alpha_1^2 \ldots \alpha_1^{\frac{l}{2}}\ \pi\alpha_1^{\frac{l}{2}}\ k\right)$$

To encrypt the dataset, $\mathcal{D}$ runs $\mathsf{FHEKeyGen}()$ to generate a public key pk and a secret key sk. Each permuted vector is then encrypted under sk and the entire encrypted data set is released to researchers, along with pk. In the scheme we use, the evaluation key is the same as the public key, but if another scheme with a separate evaluation key were to be substituted, the evaluation key would be released to the researcher instead.

4.4 Researcher Computation

Once the new data set D^* has been published, a researcher $\mathcal{R}$ prepares a transcript T of the steps of some operation they want to perform over D^*. Imagine $\mathcal{R}$ wants to compute the average age of death of patients with a certain disease who are also taking a certain medication. To compute this value, $\mathcal{R}$ uses the β associated with the disease and the β associated with the medication to include only patients with both characteristics when summing age of death.

$$\frac{\sum\limits_{\boldsymbol{d}\in D^*}(\beta_i^1 \times \beta_i^2 \times \alpha_j)}{\sum\limits_{\boldsymbol{d}\in D}\beta_i^1 \times \beta_i^2}$$

While machine learning style analysis has been growing more popular among the research community, computing more simple metrics like counts, correlations, and linear regressions are still the main methods of conducting computational analysis. All of these techniques can clearly be implemented using the same filter and sum method above. For example, a simple linear regression between the variables x_1 and x_2 can be computed as

$$x_2 = ax_1 + b$$

Such that a and b can be calculated as

$$a = \frac{\sum x_2 - b\sum x_1}{n} \; b = \frac{n\sum x_1 x_2 - \sum x_1 \sum x_2}{n\sum x_1^2 - (\sum x_2)^2}$$

where n is the number of samples in the dataset. Clearly all of these summations are easy to compute. Because of the data binning process, a researcher can also restrict their analysis to certain cohorts, focusing their attention on, for instance, subsets of the socioeconomic ladder or only more urgent hospital admittances.

4.5 Verification

When $\mathcal{D}$ receives the result of $\mathcal{R}$'s computation, he runs the verification algorithm $\mathsf{Verify}(\mathsf{T}, \mathsf{m}^*)$. It is important that the multiplicative depth of the transcript can be easily extracted; we denote the multiplicative depth d.

$\mathsf{Verify}(\mathsf{T}, \mathsf{m}^*)$ takes a transcript T and some encrypted vector c as input. The goal of the verification algorithm is to quickly decide if the steps taken in T would result in the vector c, returning 1 if it is the result vector and 0 if it is not. The verification algorithm is as follows:

1. $m = \mathsf{Decrypt}(\mathsf{c})$
2. Compute $\phi^{-1}(m)$
3. For each plaintext value a in $\phi^{-1}(m)$ make sure the corresponding verification value is $\pi^{d-1}a$, where d can be learned from T
4. Perform the computation described in T over the random tags in each vector and make sure it matches the tag of $\phi^{-1}(v)$
5. Return 1 if steps 3 and 4 both pass, otherwise return 0

While running the verification algorithm is constant in the computation time because the random tags must be computed, it is still much quicker and less memory intensive than running the computation itself. There is a single value for k in each vector, so the runtime will be at least $\frac{1}{l}$, where l is length of each plaintext vector.

If the verification algorithm returns 1, $\mathcal{D}$ strips out all values associated with the verification process before the data is put through the differential privacy process. In this way, the permutation, the random tag, and π all stay secret and the adversarial researcher gains no advantage once they perform a single valid computation. If the algorithm returns a 0, $\mathcal{D}$ must assume $\mathcal{R}$ is attempting to circumvent the encryption on the data. A cheating researcher is banned from further use of the system and their results immediately discarded.

4.6 Additive Noise

One of the goals of our construction is to make it difficult for a malicious researcher to extract information about an individual while performing a legitimate analysis. Because of the verification algorithm, we can show that it is difficult to gain information by cheating on computation. To ensure that it is difficult to gain information from legitimate analysis, we introduce differential privacy as the final step in the process. To this end, $\mathcal{D}$ adds noise sampled from a laplacian distribution with variance equal to the sensitivity of the function computed, where sensitivity is defined in Appendix C. This method has been shown to ensure differential privacy for single queries in previous works [17]. There have been no constructions for imposing differential privacy when an adversary can make any number of queries.

5 Security Analysis

It is clear that an adversarial researcher cannot directly access the plaintext data because the encryption scheme is semantically secure. We must give a heuristic argument that it is impossible for the system to leak unintended information when decrypting queries. This model is odd because it allows for limited decryption queries even though the underlying encryption scheme is not CCA2. The goal of our security analysis is to determine if it is possible for an adversarial researcher to gain information about the contents of the dataset besides the answer to the exact query specified in the transcript. Because it is difficult to characterize every kind of attack that a researcher might mount to learn about an individual in the population, we must ensure that there has been no deviation whatsoever from the supplied transcript.

In order to formalize our argument about the security of our scheme against information leakage, we begin by creating a security game. Unlike traditional games in the cryptographic setting, we do not allow an adversarial researcher to continue accessing the system once they have been caught attempting to cheat the system. In modern systems, it is common for the researcher to sign documents

making them liable for large sums of money if they are noticed attempting to recover the private information of a patient. Currently, these agreements are enforced by human log auditors. We borrow this notion and include it in our security game. The goal of the adversary is to cheat undetected; if their cheating is detected they are banned from use of the system and heavily fined (Fig. 1).

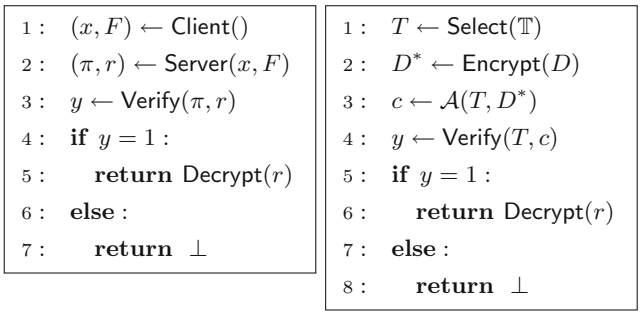

Fig. 1. Left: Traditional verifiable computation game. Right: Our updated version of this game

The traditional game for verifiable information is between a client, corresponding to the data owner, and a server, corresponding to the researcher. The client chooses some function F, usually represented in circuit form, and an input x. The server is then charged with computing $F(x)$ and proving that the computation was done honestly. We modify this game slightly to allow an adversary to select their own function, represented as a transcript, from a family of acceptable transcripts $\mathbb{T}$. We put some minimal limitations on $\mathbb{T}$, but additional limitation can be imposed by each individual data owner as needed. Valid transcripts must have the following properties:

1. The first level of computation must be performed within a single patient vector and the same computation must be performed on each patient vector.
2. The results of each such computation must be combined in a way such that the result of T when computed over the dataset is a single value (or a constant number of values with respect to the size of the dataset).
3. Results of the computation, including the processing of the results vector, must be independent of the order of vectors in the dataset.

The first property should ensure that a researcher doesn't combine β's from one patient with α's from another patient. If a researcher somehow learns about the contents of the record for a single patient and learns its location in the dataset, it should be impossible for them to leverage that information to compromise the privacy of another patient. Similarly, we require that all of the results of the computations on individual are combined into a single result. This prevents an adversarial researcher submitting a transcript that simply decrypts

patient vectors directly. Finally, the order of the vectors in the dataset should not impact the final results. Because the result of a computation over the ciphertext will yield a result vector instead of a single value, shuffling the order of the patient vectors will likely affect the individual values in the results vector but will have no impact on the sum (or product, as appropriate).

It is known to be hard to impose security policies on queries. In order to impose this specific set of security policies, the researcher is required to state their transcript in two pieces, (1) the computation to be done on each patient vector and (2) the method used to combine the results of each patient vector. Because there are no limitations on the valid kinds of computations that can be done within a single patient vector and we require that the method for combining vector results must be written in a vector independant way, any transcript that can be written in this form is valid.

We show that with our construction, the probability of creating a transcript T and ciphertext m^* that verify but were not generated honestly is bounded by the probability of guessing the random permutation Φ, specifically the location of the random tag k in the permuted vector. We assume the adversary has submitted a transcript-message pair which passes the verification algorithm, specifically recomputation of T over the random tags only. One of two things must be true: (1) the computation was done honestly or (2) some of the vectors used in the computation were altered. In the first case, clearly there is no unintended information leakage; only the answer to the adversaries exact, legitimate query has been decrypted. If some of the vectors were altered, there are two possibilities.

1. In a given vector $j < |v|$ values of the vector were altered. Given that Φ is unknown to the adversary

$$\begin{aligned} &Pr[\text{Successful Edit of } j \text{ elements}] = Pr[\text{Editing } k] \\ &+ Pr[\text{Not editing } k] \ \times \ Pr[\text{Edit results in format}] \\ &= \frac{j}{|v|} + \frac{|v|-j}{|v|}\left(1 - \frac{\binom{|v|}{j}}{\binom{\frac{|v|}{2}}{\frac{j}{2}}}\right) < \ Pr[\text{guessing location of } k] \end{aligned}$$

2. All values in some vector were edited without editing the tag k. In the worst case, an adversary has all elements of the vector besides k properly formatted (i.e. the contents of another vector in the dataset). The probability of switching out the contents of vector with the contents of another without editing the k is $\frac{1}{|v|}$.

Therefore, in all cases, the probability of an adversarial researcher computing some m without following T properly is bounded by the probability of finding k in the randomly permuted vector. We assume that the length of the vector is roughly around 1000, so $\frac{1}{|v|} \approx \frac{1}{1000}$. If this probability of being caught is too low in the eyes of the data owner, additional k's can be added to the vector. Each additional k must also be avoided when editing an existing vector, so the

chance of correcting identifying all k's in the vector goes down by approximately a multiplicative factor of $\frac{1}{1000}$ for each additional k.

6 Implementation

We implemented the above construction to measure its practical feasibility. To ensure that a medical dataset could be meaningfully transformed into the proper format, we processed NY State's public hospital discharge dataset from 2012 [32]. The dataset comprises 2.5 million patient encounters, recording data including facility information, patient demographics, patient complaint, and medical code information. While our system can scale to be used with datasets of this size, discussions with members working in the medical dataset analysis indicated that most researcher do analysis on smaller datasets of around 100,000 patient vectors. In order to test the practicality of our system, we chose to test on this *normal* cohort size.

The NY State dataset contained data in the form of text, integers and real-valued numbers. We transformed the dataset into the format described in Sect. 4.2. Some fields, like *length of stay* and *total charges*, mapped cleanly into the construction; while there were minor choices to be made regarding the granularity of the bins and how we wanted to round the decimal values to integers, the process was very intuitive. Other fields, like *admit day of week* and *APR risk of mortality* were less obvious. We chose to map each day of the week to a separate β value. In the original dataset, *APR risk of mortality* was assigned values like "Minor" and "Major". We chose to create a scale such that the lowest rating was a 0 and then each increasing level of risk was one above the previous level. Additionally we mapped each possible value of this field to its own β value. Through this process, the initial dataset, which was 100,000 vectors of length 39, was transformed into a dataset in which each vector was 912 elements long.

We encrypted large portions of the dataset for testing purposes. We chose not to encrypt the entire dataset because of space concerns, but we did encrypt 50 rows of the dataset for trial purposes. When stored naively, these 50 encrypted rows take a total of 752 GB, consuming approximately 7 MB per ciphertext. The key information and encryption context was stored in a separate file which was 16 GB. We can easily cut the size of the stored data by a factor of 2 using naive compression and there are other possible optimizations to make the storage scheme more efficient (see Appendix E.1).

Encryption was done on consumer grade electronics, specifically a MacBook Pro with a 2.5 GHz Intel i7 Core processor and 16 GB of RAM. The ciphertext was written out to an external storage device over USB 3, so the efficiency of the system was impacted severely by disk IO. We chose to set the maximum circuit depth to 100, which would accommodate most computations. We chose a security parameter of 80 and a prime modulo of 17389. Generating the context and secret key for the scheme took 22.8 min. Once the context was set up, we wrote it out to a file. To encrypt vectors, we read in the context and secret key, which took 19 min and then each plaintext vector took 10.4 s to encrypt. We split

the encryption onto two separate threads, the first thread encrypting α values and the second encrypting β values. In total, the encryption time of 50 vectors was 30.04 h and encrypting the entire dataset would have taken 584 h. Note that all the times recorded were when operations are performed linearly and without any optimizations (Fig. 2).

Task	Key Setup	Key Reading	Encryption	Sum per Ciphertext	Sum Total
Time	22.8m	15.2m	10.4s each	33.08s each	1.98hr

Fig. 2. Timing results

We performed a linear regression to test the runtime that a researcher might encounter. A linear regression is a simplistic metric to compute but is a method still often used by researchers today. Regressions and averages are basically the same operations; averages are computed with two sums and linear regressions are computed with four. Reading in the context and key information takes 15.2 min. Processing a single set of ciphertexts take 33.08 s, which includes multiplying an α ciphertext by a β ciphertext and summing it with a ciphertext that is a running sum of all previous vectors. We performed our computation without any parallelization, so a single sum of the linear regression took 1.98 h to compute when done naively. To compute the full linear regression, it took approximately 9.5 h when each sum was computed consecutively.

7 Discussion

In order for this system to be useful, there must be a clear economic incentive for the data owner. Specifically, it must be beneficial to use homomorphic encryption rather than simply performing analysis on local plaintext and returning results to the researcher. We can denote the time it take for the data owner to perform a some computation on behalf of the user as $t_{computation}$.

We consider the various costs associated with doing computation. In addition to the time to perform the computation itself, there is $t_{encryption}$, the total computation time required to encrypt a single ciphertext, and $t_{decryption}$, the time required to decrypt a single ciphertext. Additionally, the time to verify that a researcher has performed their computation honestly is denoted t_{verify}. We can express the cost of using this system for q queries as

$$\text{Cost}_{system} = \frac{|D|}{\frac{\ell}{2}} t_{encryption} + q t_{decryption} + \sum_{i=0}^{q} t^{i}_{verify}$$

whereas the cost of the data owner performing each query on the plaintext is given as

$$\text{Cost}_{naive} = \sum_{i=0}^{q} t^i_{computation}$$

The computational time required to impose differential privacy on the result of the analysis is consistent no matter the manner in which the result is computed so it can be ignored when comparing the costs of the two alternatives. Thus the marginal cost of system over simply performing the plaintext in the clear is given by

$$\begin{aligned}
&\text{Marginal Cost} = \text{Cost}_{system} - \text{Cost}_{naive} \\
&= \frac{2|D|}{\ell} t_{encryption} + q t_{decryption} + \sum_{i=0}^{q} t^i_{verify} - \sum_{i=0}^{q} t^i_{computation} \\
&= \frac{2|D|}{\ell} t_{encryption} + q t_{decryption} + \sum_{i=0}^{q} t^i_{computation}(\frac{2}{\ell} - 1)
\end{aligned}$$

Notice that the encryption time is a one-time cost incurred by the data owner; no additional encryption processing time is required for each new query posed by researchers. While the cost is very high, it can be amortized over many queries. In order for the data owner to be incentivized to use this system, the marginal cost of the system must be negative, that is

$$\frac{2|D|}{\ell} t_{encryption} + q t_{decryption} + \sum_{i=0}^{q} t^i_{computation}(\frac{2}{\ell} - 1) < 0$$

Intuitively, the computational savings from just doing verification instead of the full computation must outweigh the cost of decrypting the result vector. To give concrete examples for the variables above, we use the same parameters from Sect. 6. $t_{decryption}$ is a constant value no matter the query; as computation gets more complex the advantage of this system increases. With these parameters, decrypting a ciphertext will take approximately 18 min. We note that we measured decryption time using simple consumer grade electronics and a CPU. It may be possible to speed this process up using hardware accelerators [12,13]. In Fig. 3 we graph the marginal cost per query as a function of the decryption time and computation time, ignoring the initial encryption time. Red areas of the surface represent values for which the system is more efficient than the naive strategy. We note that the efficiency of Fully Homomorphic Encryption Schemes is likely to increase in the future, whereas the statistical tests researchers want to perform will only grow in complexity.

Remember that $t_{computation}$ denotes the total time that it would take the data owner to perform analysis, including system overhead like accessing data, which can become logistically complicated. Using this system for simple operations on small numbers of records is actually more computationally intensive for the data owner; the computation required to decrypt the results vector would be more

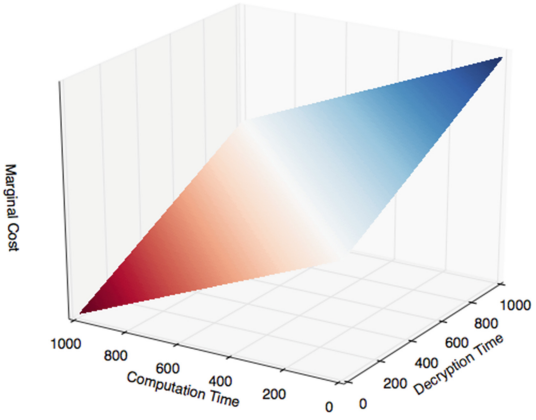

Fig. 3. Marginal cost as a function of computation time and decryption time. Negative values, red, show where this system has advantages over the naive approach (Color figure online)

than the computation itself. More complex regression methods and statistical tests are the best candidate operations for which a data owner would gain an advantage by using this system. Specifically, functionalities that would take more than the approximately 18 min decryption time. One concrete example that fits into this category is computing maximum likelihood estimators (MLE) for large numbers of parameters over a fairly large datasets. While computing simple estimators can be faster than decryption time, computing complicated estimators or estimators when data independence cannot be assumed is far more expensive. Without data independence, computing even a single iteration of MLE can be computationally infeasible on consumer hardware. Extreme examples of these costly functions can be seen in the field of economics, like [10]. While simple functions like linear regression might be the most common tools for medical researchers today, the field is growing increasingly computationally complex and being able to outsource the computation of these costly functions to researchers is a powerful tool.

In Sect. 1, we proposed six properties that would ensure that our system is useful, efficient, and secure. Our system was constructed to specifically address these properties, and we show that each one is satisfied.

Authenticity of Results. Fully homomorphic encryption guarantees addition and multiplication operate as though there were not encryption layer present. Because the researcher is doing the computation on internal systems, they do not have to be worried about some mistake in computation. We assume that the data owner is a trusted entity so there is no worry that the decrypted results do not correspond to the ciphertext delivered by the researcher. Therefore, we can conclude that all results from this system are authentic.

A Rich Range of Possible Analyses. We want to ensure that a researcher can perform any operations required for their analysis. Other solutions that manage to be both practical and cost efficient are lacking this property. The only limitations imposed on computation in our system are the limitations on valid transcripts. With access to addition and multiplication, most analysis techniques can be realized including basic machine learning algorithms.

Minimal Computation on the Part of the Data Owner. In order to maintain a secure system that can be helpful to the medical community, it is impossible not to incur high costs. The construction presented in this work shares that cost burden with researchers. For the purposes of this work, we restrict our interest to researchers with access to large computational infrastructure, like those with affiliations at universities or members of industrial researcher teams. This infrastructure currently cannot be leveraged because of difficulties obtaining data. In our discussions with individuals who work in the industry, they consider it a reasonable assumption that researchers will have access to large computational infrastructure. Most of the work we have done in our system can utilize many cores to speed up computation. No matter the computational requirements, most of the costs associated with computation are placed on the researcher. The verification time is $\frac{1}{1000}$ of the computation itself, so the system offloads $\frac{999}{1000}$ of the computation to the researcher, minus the time required to decrypt the result vector.

Privacy for Individuals in the Dataset. Fully homomorphic encryption allows for exporting the dataset does not compromise the security of any individual in the dataset. Fully homomorphic encryption guarantees semantic security, so no information can leak from ciphertext without access to a decryption oracle. Our limited decryption oracle only decrypts the results of computation that operates over the entire dataset, meaning that it can only disclose meaningful information about individuals if the legitimate query only operates over a very small subset of the dataset population. When this is the case, the noise added by the differential privacy mechanism makes it impossible to glean any information.

The main concern when sharing data is that an individual's privacy is compromised and differential privacy make that impossible for a single query. While differential privacy makes it impossible for a single query to reveal any information about a single individual in the population, it is still theoretically possible for a determined researcher to learn about an individual because we allow for multiple queries. Unfortunately, there are no constructions that we are aware of that allow for both a rich, repeated query space and multiple query differential privacy. The notion of a privacy budget, in which a researcher has a maximum number of queries or an upper bound on the allowable complexity of queries, might be used to protect about this kind of attack. We choose to leave it to each data owner if and how they would like to implement a privacy budget.

Security Against Adversarial Researchers. Because we give researchers access to a decryption oracle, it must be impossible for an adversarial researcher to simply decrypt arbitrary ciphertext. Clearly, an insecure decryption oracle would allow an adversary to trivially learn private information about individuals. The verifiable computation scheme embedded into the system guarantees that only decryption queries that operate over the entire dataset are processed. We have argued in Sect. 5 that it is very unlikely for an adversarial researcher to go unnoticed. Indeed, the data owner can tweak the probability of catching a researcher until they are comfortable with the odds.

In a traditional security model, the probability of catching a cheating adversary in our system is insufficient. Importantly, in our system a cheating adversary is banned from ever using the system again and is heavily fined. Banning an adversary prevents them from searching Φ for the location of k. Charging them for attempting to cheat means it is impractical to run multiple analyses under different identities. If there are two k's in each vector, the probability of a cheating researcher of not being caught is $\frac{1}{10^6}$, which may be insufficient for a theoretical system but is sufficient for a practical one.

Practicality. Current systems suffer from two major time related weaknesses. The first is that it takes a long time to actually begin computation. Second, if a data owner instead chooses to leverage an analysis as a service style solution, it becomes more difficult and time consuming for a researcher to access the data.

While fully homomorphic encryption does make running a single analysis significantly slower, it is important to remember that the vast majority of a researcher's time is not spent running their program. Most of the life of a research project is spent waiting to acquire a dataset or waiting to access a dataset. Our system requires a one-time cost of formatting and encryption and every future researcher will be able to use the same version of the dataset without waiting. Because we construct a system that reduces the wait time required to access a dataset, increasing the time it would take to actually perform the computation is acceptable. Recall that our goal was to make the entire process of doing research quicker, not the computation itself.

Our system also allows a researcher to perform their analysis on their own schedule. While working on this project, we found a researcher who waited months to get permission to use a specific dataset and was only able to run analysis from 2 am until 8 am while the servers storing the data were not in use; these kinds of limitations make research impossible. In our system, computation can begin without ever interacting with the data owner.

8 Conclusion

In this work we have presented a practical system for securely outsourcing medical dataset analysis. The system ensures that the researcher has the freedom to compute a rich range of metrics over the database and get results perturbed by

the minimum amount of noise to guarantee differential privacy. Our construction moves the burden of cost onto the beneficiaries of the analysis and also shortens the amount of time it takes for them to acquire and analyze a dataset. Together, these properties provide the alternative the medical research industry needs to properly incentivize data owners to share their datasets.

A Architecture Diagram

See (Fig. 4).

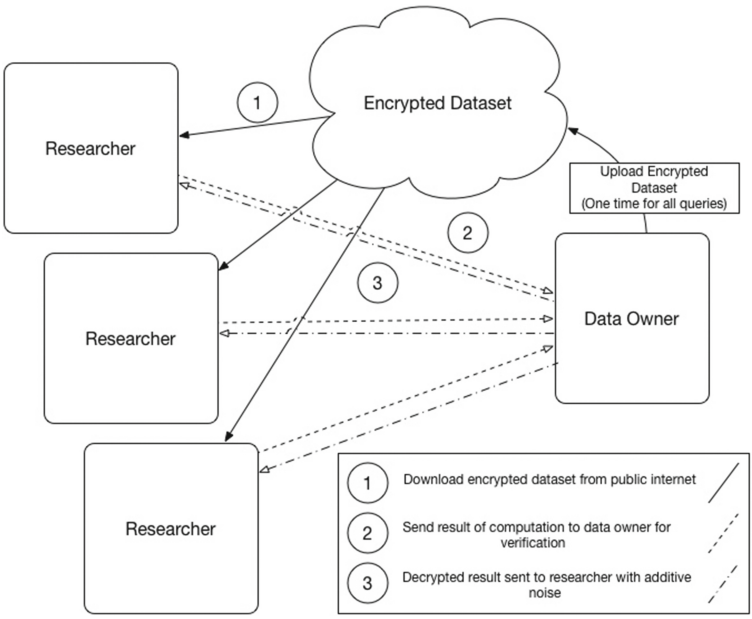

Fig. 4. System overview

B Protocol Diagram

See (Fig. 5).

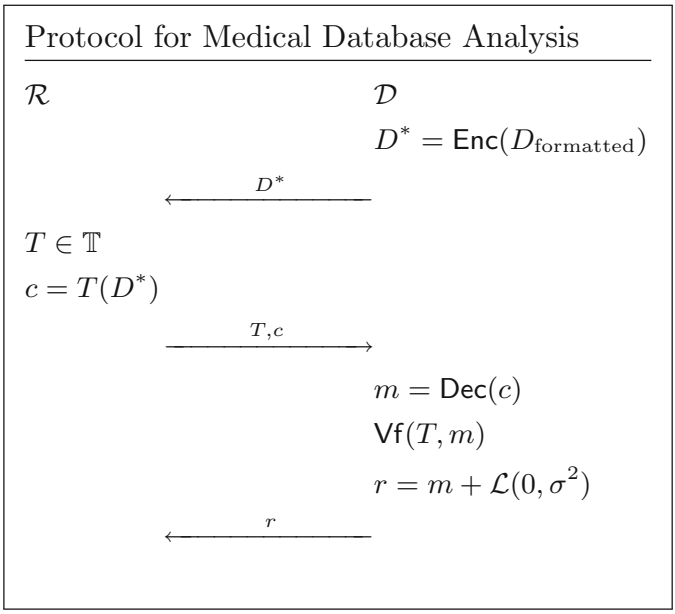

Fig. 5. Protocol diagram

C Differential Privacy

There are a number of different formal definitions for differential privacy, we choose to use the most common definition from [15–18].

Definition 1. *A randomized function $\mathcal{K}$ gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(\mathcal{K})$*

$$Pr[\mathcal{K}(D_1) \in S] \leq e^{\epsilon} \times Pr[\mathcal{K}(D_2) \in S]$$

Intuitively this means that an adversary with access to arbitrary auxiliary information can not use the function $\mathcal{K}$ to distinguish if the dataset in question is D_1 or D_2. Because D_1 or D_2 differ in at most one element, an adversary learns the same information about an individual no matter if they are in the dataset or not. Obviously a dataset without the individual contains no information about that individual, so an adversary can also learn nothing from a dataset with all information about that individual.

The most practical methods for imposing differential privacy on functions without completely destroying the usefulness of their results is introducing noise. A number of attempts have been made to create noiseless differential privacy in [6,14], but neither solution proves robust enough for our purposes. Additionally, a summary of alternative differential privacy methods can be found in [25]; we chose our solution for its elegance and computational simplicity. The most effective way to introduce noise is to add it in once the entire computation

has finished; if noise is added to the underlying data before computation, the effects of the noise are harder to predict and control [1]. Because the noise used is additive, this means that any noise-based differential privacy is secure against only single queries. If many queries are allowed, the additive noise can be cancelled out by taking an average over the multiple results. Because it is hard to decide if two queries are equivalent, protecting against these attacks is usually implemented with a privacy budget, in which only a certain number of queries are allowed for each researcher. In our construction, we do not address the issue of a privacy budget and if cancelling out the additive noise is concerning to a data owner, they should implement a privacy budget as appropriate.

C.1 Sensitivity

The method we choose for adding differential privacy to our system is adding noise sampled from a laplacian with variance equal to the sensitivity of the function computed, where sensitivity is defined as

Definition 2. *For $g : S \to R^k$, the sensitivity of g is*

$$\Delta g = \max_{S_1, S_2} ||g(S_1) - g(S_2)||_1$$

for all datasets S_1, S_2 differing in at most one element.

Because the noise is related directly to the maximum change that changing a single vector could have on the function g, it is intuitive that this method would introduce differential privacy. Computing the sensitivity of a function, at least for the class of functions relevant to this work, can be done in constant time with respect to the function itself.

Because the space of computation is limited by the transcript $\mathcal{T}$, it is easy to compute the sensitivity of any valid function. The limitations on transcripts are formalized in Sect. 5. The data owner stores a patient vector with maximum values in each α entry and a patient vector with minimum values in each α entry. Both of these vectors have all β values set to 1. The main limitation on transcripts is that the same computation is done to each vector. If we denote this computation $g(\cdot)$, the sensitivity can be computed as $|g(v_{\text{maximal}}) - g(v_{\text{minimal}})|$.

D Related Solutions

D.1 Data Simulation

One current alternative solution to anonymization and analysis as a service is data simulation. While real datasets contain information about real individuals, it is possible to construct synthetic datasets that contain no actual people but contain the same trends as a real data set. These synthetic datasets can then be released to the public without fear of compromising the privacy of any of the original patients. This is a common practice particularly in genetics research [36].

Data simulation provides an interesting solution to the same problem we are attempting to address but ultimately limits the creative abilities of researcher. Because the data is generated using statistical methods and machine learning, it is inherently limited by the foresight of its creators. The data is generated by trends observed by the data owner, but if some trend is missed, the resulting dataset will clearly not contain that trend. For this reason, synthetic data offers a wonderful opportunity to confirm previous findings but is not the best way to allow researchers to find some new information.

D.2 Verifiable Delegation of Computation

Verifiable computation or delegation of computation is a rich field of research in computer science in which a client wants to outsource some computation to an untrusted server. Because the server is an untrusted entity, the client must be able to verify that the server has done the computation honestly. In general, the problem assumes that the client has insufficient computational power to perform the original computation so the verification algorithm must be less computationally intensive than the original computation.

While there are many verifiable computation and delegation of computation constructions that we could use in our system, including [3,5,11,20,24,27], there are many requirements that are different for our problem than the traditional verifiable computation problem. Firstly, in the traditional problem there are no bounds on the computational abilities of the server; constructions prioritize lowering the asymptotical complexity of the verification algorithm at the cost of the running time of the server. Modern methods have found polylogarithmic verification algorithms, but in general the runtime of the server is completely impractical. Because we aim to construct a system that is feasible to use for both researchers and data owners, we attempt to balance the runtime on the two system such that neither is unreasonable.

Traditional Solution. The classic strategy for constructing a solution to the verifiable computation problem involves generating many function inputs, all of which look like they were selected from the same distribution [22]. One of these inputs is the true input and the others are random inputs for which the output is known to the client. The server computes the function over all the inputs and returns them all to client. If all the known outputs match the previously known outputs, the client accepts the unknown output. Otherwise, the client rejects the output and knows that the server is untrustworthy.

The obvious problem with this solution is that it requires the client to know many input - output pairs. Moreover, each time the client wants the server to perform a new computation, a new set of dummy inputs is required. Clearly this is not sustainable for a system that needs to be operational long term. Moreover, we want the server to be able to select their own circuits to compute, as a researcher in this work does. This model does not easily extend to accommodate this stipulation.

PCP and SNARKs. Modern solutions to the verifiable computation problem leverage the PCP theorem to create proofs of computation that can be checked in polylogarithmic time. Probabilistically Checkable Proofs and Probabilistically Checkable Arguments [26] are powerful cryptographic tools that allow a verifier to probabilistically sample small parts of a proof and still be convinced of its reliability. Recently, projects like [29,30,34] make the first steps towards usable PCP constructions but still fall short of being practical tools. While verification of a proof can be done quickly, the process of constructing the proof is prohibitively slow. Because one of our goals was to create a system that could be practically usable, we chose to not use any kind of PCP. As the constructions of these proofs get more efficient, it may become practical to use them instead of the proof embedding and verification methods we use in our construction.

Succinct non-interactive arguments of knowledge, or SNARKs, are an extension of zero-knowledge proofs that do not require interaction [4,7]. SNARKs allow a verifier to be convinced that a challenger possesses a witness to some NP statement without revealing the witness itself. Critically, they allow it to be done without interaction. While the zero-knowledgeness property of a SNARK would not be easily utilized, SNARKS provide another possible way to prove work. Unfortunately, SNARKs suffer from similar weaknesses as PCPs and are not practical enough for use or rely upon non-standard assumptions.

It is worth noting that the computation done by the researcher is assumed to be polynomial in the size of the dataset. If we allow the researcher to be able to compute circuits that are exponential in the size of the dataset, PCPs and SNARKS may be the only viable solutions as our verification algorithm is proportional to the computation time. Additionally, giving the researcher exponential computational power would be problematic given that the security parameter of the FHE scheme is almost certainly going to be smaller than the size of the dataset.

D.3 Systems with Limited Analytics

The work that does the best job addressing the issues with medical research is by Fiore et al. [19]. Their work creates a set of protocols to do verifiable computation on a limited set of functions computed over BGV encrypted data. They use the classic verifiable computation model in which a trusted client supplies both the encrypted data and the function F to be computed. They introduce the notion of a homomorphic, collision resistant, one-way hash function that allows the client to quickly verify if the untrusted party correctly computed F. They are able to guarantee amortized verification time that is either linear or constant in the time of computation. They are able to create protocols for performing a number of helpful functions, including linear combinations and multivariate polynomials of degree two.

While this work provides solutions to the issues of privacy, practicality and allows for the outsourcing of cost, it does not provide the flexibility required by medical researchers. While their scheme is more efficient for linear combinations, the limitations of only being able to compute multivariate polynomials of degree

two or univariate polynomials of higher degree renders their construction unsuitable for the needs of researchers. The examples cited in our implementation were already using higher order multiplication than would be supported in their work and our examples are still reasonably simple.

Another similar work is [35], in which the authors investigate the practicality of calculating statistical metrics over encrypted data. Their results of overall positive and similar to our findings. Additionally, the space of operations in their experiments are similar to our experiments. While this work provides a good start towards outsourcing medical analysis, they lack verifiable computation, a critical component given an untrusted researcher.

D.4 Personalized Medicine

Some work has been done utilizing the analysis as a service model for personalized medicine, in which a patient uploads their data to a service provider to learn some metric about their health. In [8] a system for using homomorphically encrypted data to allow the owner of a proprietary algorithm to compute a patient's risk of heart disease without learning about the patient. A similar system is [2], in which medical units can access genomic, clinical, and environmental data to compute risk metrics for a patient. The computational requirements from the FHE scheme for this problem setting are far lower than in our problem setting. The circuits computed are of lower complexity and the number of datapoints are fewer. Most importantly, computation in these system are done only over a single patient's information making the threat vectors different.

E Optimization and Future Work

E.1 Ciphertext Compression

We utilize the ciphertext I/O in the HELib to write our ciphertexts to file. While HELib provides efficient ciphertext operations, it stores its ciphertexts extremely inefficiently. The coefficients on the polynomials are all written as ascii numbers separated by spaces. To store ciphertext more efficiently, these coefficients can be stored in some binary form and then compressed. We chose to store all ciphertext in a single file for simplicity, but to minimize the size of the file a researcher would have to download, all ciphertexts containing values from a given row of the dataset should be stored in a compressed file. This storage scheme allows a researcher to pick and choose exactly what subset of the data is important to their query.

E.2 Multithreading

Much of the computation done by the researcher can be completely parallelized. Because the same operation must be done on each patient vector before the results of those computations can be combined, each of the vector operations

must be completely independent. When illustrating the viability of our system, we did no parallelization whatsoever, so all timing results are worst case. To optimize efficiency, each set of ciphertext can be processed in parallel and then combined pairwise in a tree structure. Additionally, the one-time cost of encrypting the dataset can also benefit from parallelization. Each row in the dataset can be formatted and encrypted independently.

E.3 Future Improvements to FHE

The efficiency of this system is directly tied to the efficiency of the underlying FHE scheme. We have seen tremendous strides in the efficiency of FHE since its initial construction in 2009. While we cannot anticipate the rate at which FHE will improve, it is reasonable to assume that we will see better constructions of FHE in the near future. Although we leverage the Smart-Vercauteren vectors in our construction, if future implementations do not support SIMD ciphertext operations, similar strategies can be used to bind many plaintext values together so verification can be done quickly.

References

1. Adam, N.R., Worthmann, J.C.: Security-control methods for statistical databases: a comparative study. ACM Comput. Surv. **21**(4), 515–556 (1989)
2. Ayday, E., Raisaro, J.L., McLaren, P.J., Fellay, J., Hubaux, J.-P.: Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. Presented as part of the 2013 USENIX workshop on health information technologies, USENIX, Berkeley, CA (2013)
3. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. Cryptology ePrint Archive, Report 2013/469 (2013). http://eprint.iacr.org/
4. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: Snarks for C: verifying program executions succinctly and in zero knowledge. Cryptology ePrint Archive, Report 2013/507 (2013). http://eprint.iacr.org/
5. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. Cryptology ePrint Archive, Report 2011/132 (2011). http://eprint.iacr.org/
6. Bhaskar, R., Bhowmick, A., Goyal, V., Laxman, S., Thakurta, A.: Noiseless database privacy. Cryptology ePrint Archive, Report 2011/487 (2011). http://eprint.iacr.org/2011/487
7. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. Cryptology ePrint Archive, Report 2011/443 (2011). http://eprint.iacr.org/
8. Bos, J.W., Lauter, K., Naehrig, M.: Private predictive analysis on encrypted medical data. Technical report MSR-TR-2013-81, September 2013
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) 3rd Innovations in Theoretical Computer Science, ITCS 2012, Cambridge, Massachusetts, USA, 8–10 January 2012, pp. 309–325. Association for Computing Machinery (2012)

10. Christakis, N.A., Fowler, J.H., Imbens, G.W., Kalyanaraman, K.: An empirical model for strategic network formation. Working Paper 16039, National Bureau of Economic Research, May 2010
11. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption. Cryptology ePrint Archive, Report 2010/241 (2010). http://eprint.iacr.org/
12. Cousins, D., Rohloff, K., Sumorok, D.: Designing an FPGA-accelerated homomorphic encryption co-processor. IEEE Trans. Emerg. Top. Comput. (2016)
13. Dai, W., Sunar, B.: cuHE: a homomorphic encryption accelerator library. In: Pasalic, E., Knudsen, L.R. (eds.) BalkanCryptSec 2015. LNCS, vol. 9540, pp. 169–186. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29172-7_11
14. Duan, Y.: Privacy without noise. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, New York, NY, USA, pp. 1517–1520. ACM (2009)
15. Dwork, C.: Differential privacy: a survey of results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79228-4_1
16. Dwork, C., Lei, J.: Differential privacy and robust statistics. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing, Bethesda, Maryland, USA, May 31–June 2, 2009, pp. 371–380. ACM Press (2009)
17. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_14
18. Dwork, C., Smith, A.: Differential privacy for statistics: what we know and what we want to learn. J. Priv. Confidentiality **1**, 135–154 (2009)
19. Fiore, D., Gennaro, R., Pastro, V.: Efficiently verifiable computation on encrypted data. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, New York, NY, USA, pp. 844–855. ACM (2014)
20. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. Cryptology ePrint Archive, Report 2009/547 (2009). http://eprint.iacr.org/
21. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing, Bethesda, Maryland, USA, May 31–June 2, 2009, pp. 169–178. ACM Press (2009)
22. Golle, P., Mironov, I.: Uncheatable distributed computations. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 425–440. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_31
23. Halevi, S., Shoup, V.: Helib. http://shaih.github.io/HElib/
24. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_15
25. Ji, Z., Lipton, Z.C., Elkan, C.: Differential privacy and machine learning: a survey and review. CoRR, abs/1412.7584 (2014)
26. Kalai, Y.T., Raz, R.: Probabilistically checkable arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 143–159. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_9
27. Narayan, A., Feldman, A., Papadimitriou, A., Haeberlen, A.: Verifiable differential privacy. In: Proceedings of the Tenth European Conference on Computer Systems, EuroSys 2015, New York, NY, USA, pp. 28:1–28:14. ACM (2015)

28. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy, SP 2008, pp. 111–125. IEEE (2008)
29. Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: nearly practical verifiable computation. Cryptology ePrint Archive, Report 2013/279 (2013). http://eprint.iacr.org/
30. Setty, S.T.V., McPherson, R., Blumberg, A.J., Walfish, M.: Making argument systems for outsourced computation practical (sometimes). In: ISOC Network and Distributed System Security Symposium - NDSS 2012, San Diego, California, USA, 5–8 February 2012. The Internet Society (2012)
31. Smart, N., Vercauteren, F.: Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133 (2011). http://eprint.iacr.org/2011/133
32. SPARCS: Hospital inpatient discharges (sparcs de-identified) (2012). https://health.data.ny.gov/Health/Hospital-Inpatient-Discharges-SPARCS-De-Identified/u4ud-w55t
33. Vaikuntanathan, V.: Computing blindfolded: new developments in fully homomorphic encryption (tutorial). In: Ostrovsky, R. (ed.) 52nd Annual Symposium on Foundations of Computer Science, Palm Springs, California, USA, 22–25 October 2011, pp. 5–16. IEEE Computer Society Press (2011)
34. Wahby, R.S., Setty, S., Howald, M., Ren, Z., Blumberg, A.J., Walfish, M.: Efficient ram and control flow in verifiable outsourced computation. Cryptology ePrint Archive, Report 2014/674 (2014). http://eprint.iacr.org/
35. Wu, D., Haven, J.: Using homomorphic encryption for large scale statistical analysis (2012)
36. Yuan, X., Miller, D., Zhang, J., Herrington, D., Wang, Y.: An overview of population genetic data simulation. J. Comput. Biol. **19**(1), 42–54 (2012)