Privacy-Preserving Prescription Drug Management Using Fully Homomorphic Encryption

Aria Shahverdi
University of Maryland

Ni Trieu
Oregon State University

Chenkai Weng
Northwestern University

William Youmans
University of South Florida

December 2019

Abstract

In an effort to combat the ongoing opioid epidemic in the U.S. many states have adopted a Prescription Drug Management Program (PDMP). In most cases this program has evolved into central state-wide databases for storing sensitive patient prescription records, intended for use by health care professionals to make more accurate prescribing and dispensing decisions per patient. We outline the security and privacy concerns that arise and propose a solution using privacy preserving machine learning and fully homomorphic encryption.

1 Introduction

According to the CDC [8], 46 people die every day from overdoses involving prescription opioids. In an attempt to reduce abuse of controlled medications like opioids many states implement a Prescription Drug Management Program (PDMP). The program is realized as a central database accessible to healthcare providers who can query for a record of a patient's most recently dispensed controlled medications. This allows providers to make more intelligent decisions about when to prescribe or dispense these medications. It also helps prevent "doctor shopping", where a patient sees multiple doctors for the same condition to obtain more controlled medications, either with the intention of abusing or redistributing the medication.

Currently, patient data in this system can be accessed by pharmacists, physicians, insurance companies, law enforcement agencies, and others. After registering with the program users are given access to all patient records. Granting access to such sensitive records to so many parties is an obvious security concern, and it comes as no surprise that a breach has already occurred. In Florida in 2013 [10], over 3000 patient records were shared with county prosecutors as part of a criminal investigation, when many were irrelevant to the case. An attorney involved in the investigation discovered a friend on the list and gave them the data so they could pursue legal action.

We propose a solution utilizing Privacy Preserving Machine Learning (PPML) and Fully Homomorphic Encryption (FHE) to preserve the benefits of the PDMP database while eliminating the risk of leaking sensitive patient information. We accomplish this by transferring control of the data back to the patient. Patient records will be stored in a central server and encrypted using

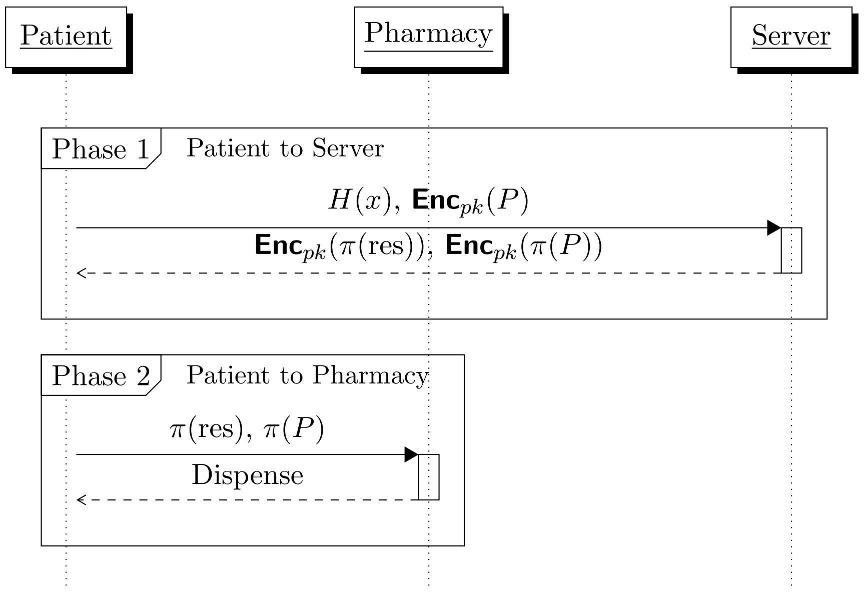
Patient
Pharmacy
Server
Phase 1
Patient to Server
$H(x)$, $\mathsf{Enc}_{pk}(P)$
$\mathsf{Enc}_{pk}(\pi(\mathrm{res}))$, $\mathsf{Enc}_{pk}(\pi(P))$
Phase 2
Patient to Pharmacy
$\pi(\mathrm{res})$, $\pi(P)$
Dispense

Figure 1: The overview of our scheme.

the patient's own key. Prescribers or pharmacists can submit a patient's encrypted prescription requests to this central server. Here, thanks to advances in FHE, we can perform PPML on the patient's encrypted data to produce an encrypted certificate authorizing or denying the prescribing or dispensing of the medication based on their history. Once decrypted, this certificate can be authenticated by the healthcare provider to prevent potential tampering or counterfeiting. Finally, the server can update the patient's encrypted record without the need for decryption by using the properties of FHE. This protocol will prevent the use of the database for any other purpose than determining if a patient is eligible for a controlled medication.

2 Our Model

For simplicity we will describe a scheme which only considers three parties: the patient, pharmacist, and server. We assume the server holds a database of patient records – each encrypted under a separate key – as well as a machine learning model trained to detect potential abuse of medication. In practice there are other parties that might want to be involved, such as prescribers that want to ensure a patient is not potentially abusing their medication. The scheme described below can easily be extended to account for this scenario.

Since we desire that each patient encrypts their data with their own key, we need a method of ensuring that encrypted responses from the server decrypted by the patient and delivered to the pharmacy have not been tampered with. We propose that the server and pharmacy create a shared secret key for some symmetric scheme that can be efficiently evaluated homomorphically for our choice of FHE scheme. The details of the FHE scheme, PPML model and symmetric scheme for authentication will be discussed in Section 3.

The overview of our scheme is presented in Figure 1. It consists of two phases: 1) submitting a prescription request, where a PPML model determines if the medication is safe to dispense, and 2) authentication of the result in the pharmacy. The first phase can be done entirely between the patient and the server, see Figure 2. Consider an example where a patient has a prescription for a controlled medication. They send to the server the hash of their identity $H(x)$ as well as the encryption $\textsf{\textbf{Enc}}_{pk}(P)$ of their prescription details (either scanned or manually entered, for example). The server evaluates the PPML model on the patient's existing encrypted record $\textsf{\textbf{Enc}}_{pk}(R)$ and new encrypted prescription to produce an encrypted result $\textsf{\textbf{Enc}}_{pk}(\text{res})$ authorizing or denying the new prescription based on the patient's history. We will refer to this result of the model as the "label".

INPUT: Hashed patient identifier $H(x)$, encrypted patient prescription $\textsf{\textbf{Enc}}_{pk}(P)$.

- Locate the patient's encrypted record $\textsf{\textbf{Enc}}_{pk}(R)$ using their identifier $H(x)$.
- Obtain the output $\textsf{\textbf{Enc}}_{pk}(\text{res})$ of the PPML model with input $\textsf{\textbf{Enc}}_{pk}(R)$ and $\textsf{\textbf{Enc}}_{pk}(P)$.
- Homomorphically compute $\textsf{\textbf{Enc}}_{pk}(\pi(\text{res}))$, $\textsf{\textbf{Enc}}_{pk}(\pi(P))$ for a symmetric scheme π.

OUTPUT: Send to the patient $\textsf{\textbf{Enc}}_{pk}(\pi(\text{res}))$, $\textsf{\textbf{Enc}}_{pk}(\pi(P))$.

Figure 2: Patient-server interaction in phase 1.

In the second phase, the patient decrypts the server response to obtain $\pi(\text{res})$ and $\pi(P)$ which they provide to the pharmacist. Optionally, the server can return $\textsf{\textbf{Enc}}_{pk}(\text{res})$ as well, if it's desired that the patient sees the label at this step. The pharmacist decrypts $\pi(\text{res})$ and $\pi(P)$ with the shared secret key to learn the result of the model as well as verify that the prescription data P sent to the server matches the prescription brought to them by the patient. We discuss how to prevent tampering or a dishonest patient from providing an incorrect or previously used label in Section 3.3.

Lastly, the server updates the patient's encrypted record with the new encrypted prescription data. We can assume the new prescription data is stored from the initial round of communication or sent again by the pharmacy. In either case, the technique for merging encrypted records is covered in Section 3.1.

3 Details

3.1 Fully homomorphic encryption

There are three major aspects of our design which influence our choice of FHE scheme:

1. The server must be able to make predictions on a patient's encrypted data.
2. Healthcare providers will need a method of authenticating the server response to prevent tampering by the patient.
3. Upon dispensing a medication the patient's record will need to be updated.

3.1.1 Our choice of FHE scheme

In order to accommodate all of these needs we chose to use the Brakerski-Gentry-Vaikuntanathan (BGV) FHE scheme as described in [2]. While other schemes can handle requirements 1 and 3, the authentication step mentioned in requirement 2 means we will need an FHE-friendly symmetric encryption scheme. LowMC [1] is a block cipher with comparatively low multiplicative depth designed for use with FHE and multi-party computation. Lattice-based symmetric encryption schemes based on Learning With Errors (LWE), Learning Parity with Noise (LPN) and Learning With Rounding (LWR) were described in [5] and provide another option. All of these shcemes were demonstrated to be efficient to evaluate homomorphically using BGV as implemented in HElib [7]. The details of this will be discussed in Section 3.3.

The next most important requirement is that we can homomorphically evaluate the machine learning model. For the most accurate models this means approximating non-linear functions in a manner suitable to homomorphic operation. We give a more detailed survey of the potential solutions in Section 3.2.

3.1.2 Updating the encrypted records

The last usage of our FHE scheme is updating patient records. For this we propose taking advantage of the ciphertext packing techniques of Smart and Vercauteren [9] which allow for single instruction multiple data (SIMD) operations. With this we can perform operations on a single ciphertext which translates to performing parallel operations coefficient-wise on a vector of plaintext slots. We also use homomorphic rotations of ciphertexts which corresponds to rotating the underlying plaintext slots. Both techniques are available in the current implementation of BGV in HElib.

Consider a simplified example where a patient prescription contains only the drug name, quantity, and date. We represent FDA approved drugs by their NDC or National Drug Code. We will write a patient's plaintext prescription as a vector $P = \langle n, q, d, 0, ..., 0\rangle$ of length m determined by the number of plaintext slots available with the given BGV parameters. Let n, q, and d represent the NDC, quantity, and date respectively. We will represent a patient record containing the NDC, quantity, and date of the last m prescriptions as a triple (R_1, R_2, R_3) where $R_1 = \langle n_1, ..., n_m\rangle$, $R_2 = \langle q_1, ..., q_m\rangle$, and $R_3 = \langle d_1, ..., d_m\rangle$. We will assume the records are in order from newest to oldest. A prescription encrypted under a patient's public key pk takes the form $\textsf{Enc}_{pk}(P) = \textsf{Enc}_{pk}(n, q, d, 0, ..., 0)$, and an encrypted patient record $\textsf{Enc}_{pk}(R) = (\textsf{Enc}_{pk}(R_1), \textsf{Enc}_{pk}(R_2), \textsf{Enc}_{pk}(R_3))$. Lastly write $\mathbf{0}_i$ for the plaintext vector which is 0 everywhere and 1 in the i-th slot, and $\mathbf{1}_i$ for the plaintext vector which is 1 everywhere and 0 in the i-th slot. In algorithm 1 we demonstrate a possible approach to updating encrypted patient records by rotating the ciphertexts and overwriting the m-th prescriptions data.

Algorithm 1 Update database

Input: encrypted patient prescription $\textsf{Enc}_{pk}(P)$, encrypted patient record $\textsf{Enc}_{pk}(R)$.
Output: updated encrypted patient record $\textsf{Enc}_{pk}(R')$.

```
1: for $i \in \{1, 2, 3\}$ do
2:     $A \leftarrow \textsf{Enc}_{pk}(P * \mathbf{0}_i) = \textsf{Enc}_{pk}(P) * \textsf{Enc}_{pk}(\mathbf{0}_i)$        Erase all but the $i$-th slot of $P$.
3:     $A \leftarrow \mathrm{rot}(A, m - i + 1)$        Rotate slot $i$ to slot 1.
4:     $B \leftarrow \textsf{Enc}_{pk}(R_i * \mathbf{1}_m) = \textsf{Enc}_{pk}(R_i) * \textsf{Enc}_{pk}(\mathbf{1}_m)$        Erase slot $m$ of $R_i$.
5:     $B \leftarrow \mathrm{rot}(B, 1)$        Rotate slot $m$ to slot 1.
6:     $x_i \leftarrow \textsf{Enc}_{pk}(R'_i) = A + B$        Insert the prescription data.
7: end for
8: return $(x_1, x_2, x_3)$
```

3.1.3 Parameters

Choosing parameters for use with any FHE scheme is a delicate task involving many factors, often reducing to experimentation for fine tuning. Since we currently lack an implementation determining the parameters is even more challenging. However, we note that BGV supports switching back and forth between plaintext moduli of the form 2^k via a recryption process.

We outline a general approach inspired by Crawford et al. [4]. By storing our data (prescriptions and prescription records) in packed ciphertexts with plaintext modulus 2^k for some large enough k, we can extract the k encrypted bits as necessary at the cost of some predictable amount of homomorphic capacity depending on the starting parameters. This technique allowed the authors of [4] to implement more efficient approximation of non-linear functions using homomorphic look-up tables. This also simplifies the homomorphic evaluation of the decryption circuit of a symmetric scheme that works bit-wise. In [4] BGV was used with plaintext space the m-th cyclotomic integer ring for $m = 2^{15} - 1$ and plaintext modulus 2^{11}. This corresponds to lattices of dimension $\phi(m) = 27,000$ and a recryption step costing 20 levels. This is the cost of extracting the 11 encrypted bits from each slot. They chose to use 29 levels in the BGV moduli-chain resulting in a ciphertext modulus q of roughly 1030 bits and overall security of more than 80 bits. These parameters result in 1800 plaintext slots per ciphertext, each able to contain 11 bit integers. We expect this should also be sufficient for our case, and the ability to extract encrypted bits will allow us to directly use the existing implementation of the LowMC block cipher demonstrated in [1] to homomorphically operate on encrypted bits using BGV.

3.2 The machine learning model

The machine learning model should take as input the patient's encrypted record and new encrypted prescription. Then it runs a classification algorithm resulting in a decision which tells the pharmacy whether or not they should dispense the medication. The traditional models used for prediction in the healthcare industry involve logistic regression, support vector machines (SVM), and random forests. Homomorphic encryption is not well suited to the branching computations involved in random forests. SVM requires keeping all of the training data to make predictions [3], which may or may not be suitable depending on how we choose to train the model. Logistic regression is a feasible option that is well studied in the context of BGV [4]. Neural networks are also a candidate, assuming we can approximate the non-linear functions involved in a way suitable for homomorphic operations. In any case, it will be necessary to either extract feature vectors from the encrypted patient records using the techniques of Algorithm 1 or to store patient records as feature vectors directly.

3.2.1 Training the model

Our options for training the model are severely restricted by our requirement that each patient's data be encrypted under a distinct key. Training on encrypted data in this setting is not straightforward. Instead we propose to generate anonymous patient records and prescriptions and ask health professionals to assign a label depending on if the prescription should be approved or not. This way the model can be trained on clear data and only predictions involve computation on encrypted data.

3.2.2 A remark on using ML

There are some concerns that should be taken into consideration when using machine learning in this context. Since the training data must be labeled by health professionals it will unfortunately capture any biases in their decisions. Ideally our system would not just improve privacy but also the accuracy and fairness in the decision to prescribe a patient controlled medications. To this end, it may be more efficient to encode a set of rules for approving or denying controlled prescriptions and bypass machine learning altogether. This will need to be taken into consideration in any practical applications.

3.3 Authentication

Assume that a patient has submitted an encrypted prescription $\textsf{Enc}_{pk}(P)$ to the server. The server feeds the prescription and encrypted patient record $\textsf{Enc}_{pk}(R)$ to the model, which produces a encrypted label $\textsf{Enc}_{pk}(\text{res})$. Now the server is tasked with sending this data to the patient, to be decrypted with the patient's secret key and presented at the pharmacy. It is possible that the patient may try to modify the result of the model, make up a fake result altogether, or mix and match labels and prescriptions to their advantage. In order to prevent patient tampering during this transaction, we introduce an authentication step under the assumption that the server and pharmacy have previously agreed on a shared secret key.

3.3.1 The shared secret key

One may consider letting the server sign a digital signature over the ciphertext. However, this is not enough for the pharmacy to verify the authenticity of the plaintext result. One solution is homomorphic evaluation of the decryption circuit of a symmetric-key encryption scheme. Implementations of AES [6] as well as LowMC [1] – a block cipher with low AND depth and low multiplicative complexity specially developed for use with FHE – have been developed using HElib. Lattice based schemes have also been studied in this context and show promise [5]. For this work we will use LowMC.

Assume that each pharmacy shares a secret key ss with the server, which is also the encryption key of the algorithm LowMC, $\pi()$. In phase 1 of our scheme, the server will return to the patient the output of the model as well as the encryption under the shared secret key: $\textsf{Enc}_{pk}(\text{res})$, $\textsf{Enc}_{pk}(\pi_{ss}(\text{res}))$, and $\textsf{Enc}_{pk}(\pi_{ss}(P))$. The patient then decrypts the response using the FHE secret key sk and obtains res, $\pi_{ss}(\text{res})$, and $\pi(P)$. The patient can see the output of the model themselves, and sends $\pi_{ss}(\text{res})$ and $\pi_{ss}(P)$ to the pharmacy. The pharmacy decrypts to recover the result of the model, and checks that the prescription P matches that presented by the patient.

3.3.2 Prevent patient tampering

Lastly we need to ensure the patient can not mix and match results of the model with different prescriptions. One possible solution is to preserve a plaintext slot for a serial number or timestamp, i. Then before encryption with $\pi()$, the server inserts i into the plaintext slot using the same strategy used previously to update a patient record. The patient receives $\textsf{Enc}_{pk}(\text{res}, i)$, $\textsf{Enc}_{pk}(\pi_{ss}(\text{res}, i))$, and $\textsf{Enc}_{pk}(\pi_{ss}(P, i))$. On final decryption the pharmacy can verify that the slots match. Using timestamps has the added advantage of allowing a potential expiration for the authentication. If the check passes, the pharmacy can fill the prescription or not based on the decision made by the machine learning model.

References

[1] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *Advances in cryptology—EUROCRYPT 2015. Part I*, volume 9056 of *Lecture Notes in Comput. Sci.*, pages 430–454. Springer, Heidelberg, 2015.

[2] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully Homomorphic Encryption without Bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `https://eprint.iacr.org/2011/277`.

[3] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. In *Machine Learning*, pages 273–297, 1995.

[4] Jack L.H. Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing Real Work with FHE: The Case of Logistic Regression. Cryptology ePrint Archive, Report 2018/202, 2018. `https://eprint.iacr.org/2018/202`.

[5] Pierre-Alain Fouque, Benjamin Hadjibeyli, and Paul Kirchner. Homomorphic Evaluation of Lattice-Based Symmetric Encryption Schemes. In Thang N. Dinh and My T. Thai, editors, *Computing and Combinatorics*, pages 269–280, Cham, 2016. Springer International Publishing.

[6] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. Cryptology ePrint Archive, Report 2012/099, 2012. `https://eprint.iacr.org/2012/099`.

[7] S. Halevi and V. Shoup. HElib - an implementation of homomorphic encryption, September 2014.

[8] H. Hedegaard, B.A. Bastian, J.P. Trinidad, M. Spencer, and M. Warner. Drugs most frequently involved in drug overdose deaths: United States, 2011–2016. *National Vital Statistics Reports*, 67, 2018.

[9] N.P. Smart and F. Vercauteren. Fully Homomorphic SIMD Operations. Cryptology ePrint Archive, Report 2011/133, 2011. `https://eprint.iacr.org/2011/133`.

[10] Times Staff Writer. Tampa Bay Times, 2013. `https://www.tampabay.com/news/politics/did-floridas-prescription-pill-database-really-spring-a-leak/2130108/`.