SecureLR: Secure Logistic Regression Model via a Hybrid Cryptographic Protocol

Yichen Jiang†*, Jenny Hamer†*, Chenghong Wang†, Xiaoqian Jiang , Miran Kim,
Yongsoo Song, Yuhou Xia, Noman Mohammed, Md Nazmus Sadat, Shuang Wang

Abstract—Machine learning applications are intensively utilized in various science fields, and increasingly the biomedical and healthcare sector. Applying predictive modeling to biomedical data introduces privacy and security concerns requiring additional protection to prevent accidental disclosure or leakage of sensitive patient information. Significant advancements in secure computing methods have emerged in recent years, however, many of which require substantial computational and/or communication overheads, which might hinder their adoption in biomedical applications. In this work, we propose SecureLR, a novel framework allowing researchers to leverage both the computational and storage capacity of Public Cloud Servers to conduct learning and predictions on biomedical data without compromising data security or efficiency. Our model builds upon homomorphic encryption methodologies with hardware-based security reinforcement through Software Guard Extensions (SGX), and our implementation demonstrates a practical hybrid cryptographic solution to address important concerns in conducting machine learning with public clouds.

Index Terms—Logistic Regression, Homomorphic Encryption, Secure Cloud Computing, SGX, Machine Learning

† *Y.J., J.H. and C.W. contribute equally to this paper. *Please direct all correspondence to Y.J & J.H.*

- *Y.J., J.H., C.W., X.J., M.K. and S.W. are with the Dept. of Biomedical Informatics, University of California San Diego, La Jolla, CA. E-mails: {yij152, jhamer, chw148, x1jiang, mrkim, shw070}@ucsd.edu*
- *Y.S. is with the Dept. of Computer Science and Engineering, University of California San Diego, La Jolla, CA. E-mail: yongsoosong@ucsd.edu*
- *Y.X. is with Department of Mathematics, Princeton University, Princeton, NJ. E-mail: yuhoux@math.princeton.edu*
- *M.S. and N.M. are with the Dept. of Computer Science, University of Manitoba, Winnipeg, MB R3T 2N2, Canada. E-mails: {noman, sadat}@cs.umanitoba.ca*

1 Introduction

The abundance of biomedical data [1] provides many opportunities for knowledge discovery in the healthcare and biomedical domains, and as a result, data increasingly drive biomedical research initiatives [2], [3]. However, researchers using these data hold the responsibility of ensuring their privacy and security during the entire process of data analysis. The recent influx of malicious attacks on medical data [4], [5] increases the need for improved data protection measures and to prevent sensitive information leakage to unauthorized users.

Administrative and technical safeguard solutions have been developed and improved during the past decade. Many novel approaches and policies [5]–[9], have been in place to provide protection for sensitive biomedical data. Due to the large size of biomedical data, storage and computation are so overwhelming that it is hard for small institutions to host and analyze such data efficiently. Cloud computing has emerged as an ideal enabling platform due to the scalability and cost-effectiveness. However, new challenges have emerged as a result of this practice, where currently, health data are regulated by privacy policies such as the HIPAA Privacy Rule [9]. Additionally, computational resources in the cloud are shared among different users, and data within the cloud may be backed up across different regions, which impose further privacy and security risks. When medical records are not properly protected, security breaches within the health sector may result in identity theft or medical fraud for patients who inadvertently become victims of these attacks, which can, among other adversities, cause financial devastation or lead to social humiliation and discrimination when their personal information is leaked [10]. Motivated by these challenges, it is imperative to develop secure and efficient methods to support valuable biomedical studies and to facilitate open science [11] in biomedical research.

In this paper, we focus on the study of secure predictive modelling in public cloud environments, where both data storage and analysis are outsourced. More specifically, we target the logistic regression algorithm, which is widely used in many different biomedical applications such as disease prediction, risk assessment and so on. For example, logistic regression has been shown capable of accurately predicting survival among patients with advanced liver disease [12] and to provide decision support in the early diagnosis of acute myocardial infarction [13]. There have been many existing studies contributing to the problem of secure and privacy-preserving logistic regression analyses based on techniques like differential privacy [14], [15], federated data analysis [16]–[18], and cryptographic methods [19]–[22]. In federated data analysis, it assumes that data are distributed among different institutions. Instead of directly sharing patient-level data with others, the intermediary statistics with much less sensitive information will be exchanged to build a global training protocol for logistic regression. For example, a Grid Binary LOgistic REgression (GLORE) model [16] was proposed to estimate global model parameters for horizontally partitioned data (i.e. data in each institution have the same attributes but they are from different patients). GLORE was further extended by Wang et al. [17] to support privacy-preserving federated online learning in a Bayesian paradigm. Moreover, the protection of exchanging intermediary statistics in distributed model learning were studied in [19], [20], [22] by using different crypto schemes (e.g., homomorphic encryption and secure multiparty computation). However, only limited studies have been conducted to support securely logistic regression in a cloud environment. Another relevant work is from Aono et. al. [21] which proposed a quadratic approximation method of the standard non-polynomial sigmoid function to support outsourced training tasks in logistic regression. However, such methods still require the interaction between the cloud and the data owner. Our goal is to develop a security model to support fully outsourced logistic regression analysis.

Ideally, we would like to use homomorphic encryption (HME) [23], a cryptosystem providing rigorous guarantees and being natural to secure outsourcing. However, existing HME-based techniques [24] require substantial storage and computational overhead, which make them impractical for biomedical analysis tasks [24]. On the other hand, the recently released Software Guard Extensions (SGX) architecture provides an advanced hardware isolation mechanism to enable data owners to seal sensitive data computation inside an enclave. This new feature is readily available in the consumer market in Intel (6th-generation or higher) processors. The claim is that any computation completed within the enclave can be protected from most external attacks with minimal overhead. Despite biomedical applications demonstrated by us in [25]–[27] using SGX, some recent studies [28], [29] have shown a few potential attacks for SGX-based framework, such as controlled side-channel attack, cache-timing attack, etc.

We aim to combine the best of both worlds. The main contributions of this study are threefold. First, we aim at closing this technology gap to help biomedical researchers perform fully outsourced secure logistic regression analysis efficiently by taking advantage of a hybrid solution incorporating HME and SGX based secure hardware [30] technologies [30]. Furthermore, inspired by the idea of SMC secret sharing [31], our proposed hybrid design distributes encrypted computation between SGX-based secure hardware and HME to achieve not only a strong protection but also better efficiency. Lastly, we evaluate the proposed SecureLR framework on various datasets to demonstrate its capabilities and compare its performance against a plaintext implementation and previous secure logistic regression solutions [20].

2 Background

2.1 Homomorphic Encryption

Homomorphic Encryption is an encryption scheme which allows data to undergo certain arithmetic operations (e.g. addition and multiplication) while protected under encryption, or in the form of a *ciphertext*, without requiring the data to be revealed or decrypted. Three predominant schemes are employed - *partial*, *fully*, and *leveled* homomorphic encryption (HME) - each with a trade-off between computational and space efficiency and complexity [32]. While partial HME may offer less space and computational overhead compared to the other two schemes, it only allows for a single type of operation, either addition or multiplication, between two ciphertexts. On the contrary, fully HME schemes support an unlimited number of both addition and multiplication operations, but require a time-consuming bootstrapping process to refresh the ciphertext *noise*, or the small quantity of error which accumulates during homomorphic arithmetic operations, which renders it impractical for many real-world applications. Considering these disadvantages, we leverage leveled HME, also referred to as leveled fully HME, which can support a parametrically-set number of both arithmetic operations without resulting in decryption errors [33].

HME pertains to two mathematical spaces, the plaintext and ciphertext spaces, which we discuss as follows. Assuming that n is a power of two, the plaintext space is given by the ring structure $R_t = Z_t[x]/(x^n+1)$, where t is the plaintext modulus, meaning that this space is defined by polynomials of degree less than n and with coefficients modulo t. The ciphertext space defined similarly by the ring $R_q = Z_q[x]/(x^n+1)$, where q is the coefficient modulus. It is in the ciphertext space where we perform encryption and homomorphic multiplication and addition operations. These ring structures are in part defined by the polynomial modulus $p = x^n + 1$ (as in the ring "learning with errors" or (R)LWE problem [34]). By setting q and t as prime numbers such that $q \equiv 1 (mod\, 2n)$ and $t \equiv 1\ (mod\, 2n)$, we may utilize the Number Theoretic Transform (NTT) for multiplication between polynomials, which improves computational performance. This in turn lets us take advantage of CRT batching, which allows for many distinct values to be stored in a single ciphertext upon which we apply arithmetic operations on all elements within the ciphertext in a Single Instruction Multiple Data (SIMD) manner.

Each operation under homomorphic encryption operation introduces a small bit of error, called *noise*, within the given ciphertexts. Once the ciphertext's accumulated noise reaches the maximum noise budget, its decryption is inaccurate (i.e. the encrypted information is overwritten with noise) [33]. To overcome this, bootstrapping operations are used in fully HME, however, this approach is computationally expensive. We propose a mechanism to "refresh" our ciphertexts, or remove this error by decrypting and re-encrypting in an SGX enclave as an efficient and secure alternative (see §3.4 and Algorithm 3 for further discussion).

2.2 Software Guard Extensions

Software Guard Extensions (SGX) is a security feature of the Intel (6th-generation or later) processor architecture, which provides a hardware-supported area for secure and confidential computing, called an *enclave*. SGX uses an "inverse sandbox" design method, which seals private codes, sensitive data, and other selected "secrets" into the enclave's memory, which may not be read or written to by unapproved or untrusted components regardless of CPU mode or privilege level. The enclave encapsulates confidential data and computation, and serves to prevent its leakage in the event that privileged modules become corrupted, vulnerable to attacks, or potentially malicious [35]. When lower-level or privileged components become compromised, e.g. the BIOS or Operating System (OS), applications that properly utilize an enclave may maintain their security guarantees. As shown in Figure 1, a traditional application would arbitrarily allow direct access to its data through function calls or remain susceptible to infiltration through memory manipulation. In contrast, applications supported by SGX can seal secret data within an enclave and exclusively allow access to trusted or authorized code. Therefore by using SGX, applications that undergo malicious attacks or run on potentially untrusted hosts can maintain their security integrity.

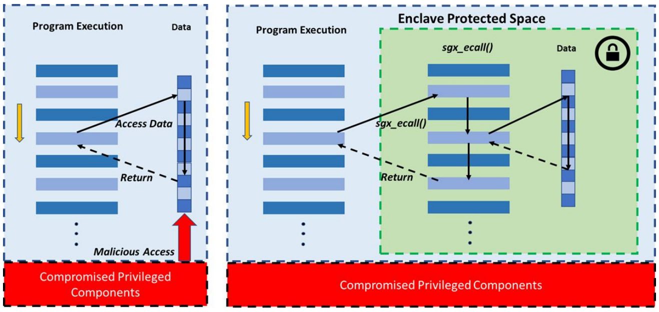


Fig. 1. Traditional vs. SGX-enabled Computation. Comparison between traditional and SGX-enabled computation models when privileged hosting components are compromised.

2.3 Logistic Regression

Logistic regression is a binary classification algorithm with wide applications in the biomedical informatics, including clinical decision support,, risk assessment, and disease classification. Logistic regression estimates the probability (i.e. *predicts)* a categorical characteristic based on a combination of input variables (e.g., estimates *1*: that a patient exhibits condition *A*, or *0*: patient does not have condition *A*).

Suppose a dataset consists of *n* records (or *samples* used for *training and testing* the regression model), and *m features* (independent or predictor variables), with an observed binary class label for each record (dependent variable). We denote the training records by a matrix $X = (x_1, x_2, ..., x_n)^T \in R^{nm}$ such that each row vector $x_i \in X$ is $x_i = (x_i^1, x_i^2, ..., x_i^m)^T$, where x_i^j is the j^{th} feature of the i^{th} record, and the corresponding labels are *Y*, where $Y = (y_1, y_2, ..., y_n)^T$. We also add an additional variable to *X* serving as a *bias*, which will represent the classification probability when all predictor variables are *0*. Then, given the likelihood function of the form

$$p(y_i = 1|x_i^T w) = g(x_i^T w) = 1/(1 + exp(-x_i^T w))$$

we can *learn* or optimize regression model parameters $w = (w^1, w^2, ..., w^m)^T$, where by training a regression model on the given *X* and *Y* through maximizing the following log-likelihood function:

$$L(w) = \sum_{i=1}^{n} y_i log(g(x_i^T w)) + (1 - y_i) log(1 - g(x_i^T w)) .$$

Because there is no closed-form solution to this function, we maximize it by approximating its gradient with respect to w, using an algorithm called *gradient ascent* (which we discuss in more detail in §3.4). Here, the logistic function, $g(z) = 1/(1 + exp(-z))$, often referred to as the *link function* in logistic regression, is sigmoidal (an S-shaped curve). As its codomain lies on the interval [0,1], it lends itself as an ideal function for computing the probability of an outcome based on the inputs at which it is evaluated. Furthermore, this form of regression does not require that its dependent variables (y_i's) be normally distributed, which makes it easy to handle binary classification problems [36].

3 Methodology and Protocol Design

Our proposed model consists of four primary entities, who each play a role in the SecureLR protocol and perform transactions contributing to its functionality and security.

1. **Data Owners (DOs)**: an institution that holds the biomedical data and would like to outsource their storage and analysis to Cloud Service Providers (CSPs) in secure manner.
2. **Cloud Service Providers (CSPs)**: trusted entities that provide: (i) the outsourced storage of data (encrypted, from DOs), (ii) the interface for Authorized Researchers (ARs) to perform secure logistic regression over such datasets, and (iii) return the trained model parameters in ciphertext to the requesting ARs.
3. **Authorized Researchers (ARs)**: individuals who receive authorization by an Authentication Service provider (ASP) to request SecureLR services through the CSP (using certain encrypted datasets from DOs). The ARs receive encrypted, trained regression parameters and may decrypt them locally.
4. **Authentication Service Provider (ASP)**: A trusted entity which performs authorization of: (i) the DOs to outsource encrypted private datasets to a CSP, (ii) the ARs to request SecureLR training services on CSP using these outsourced and encrypted data, (iii) the ARs for the decryption of encrypted model parameters, (iv) verification of the remote enclave on the CSP to ensure the security and integrity during the entire lifecycle of SecureLR.

Our proposed protocol assumes the involvement of two CSPs [37], [38], one to perform HME computation and the other to support the enclave (i.e. for secure hardware computation). We consider the two CSPs are non-colluding, semi-honest parties, where each correctly follows the protocol, but might be curious about other's information if it is directly observable.

The SecureLR framework workflow consists of five key processes (as illustrated in Fig. 2) which we discuss in detail in the following sections.

3.1 Homomorphic Encryption Initialization

Step 1: Initialize and Distribute HME Parameters (CSP1→ARs, CSP2, DOs). In the SecureLR framework, the CSP1 initializes the homomorphic environment. The current SecureLR framework is optimized for the FV (Fan-Vercauteren) HME scheme [33], however one may choose other HME schemes [39], [40]. This environment requires setting a few specific parameters prior to applying its encryption functions. These parameters impact many aspects of the framework, including its level of security, computational capacity, and size of encrypted data, and thus need to be selected carefully (see §4 for a brief description on parameter selection). Once initialized, these parameters are distributed to the DOs, ARs, and CSPs.

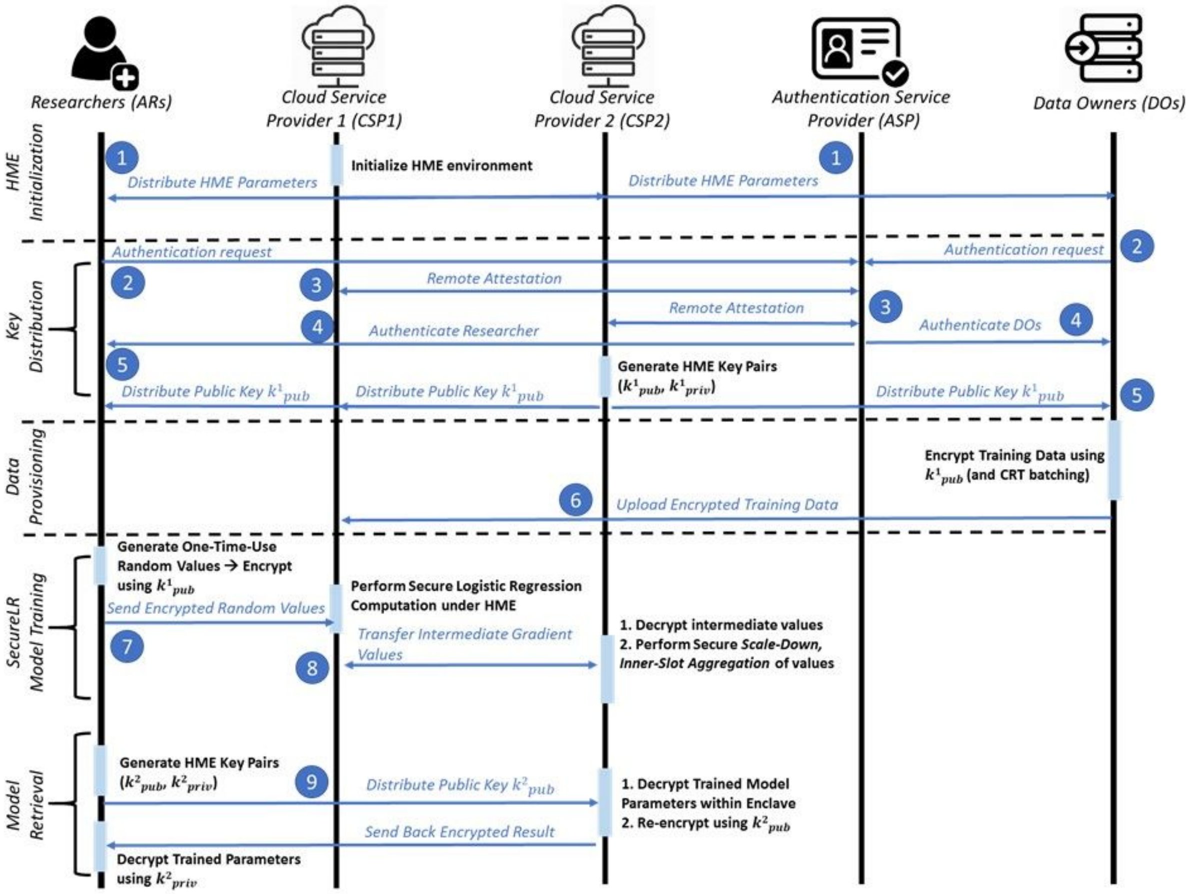

Researchers (ARs)
Cloud Service Provider 1 (CSP1)
Cloud Service Provider 2 (CSP2)
Authentication Service Provider (ASP)
Data Owners (DOs)
HME Initialization
1
Distribute HME Parameters
Initialize HME environment
1
Distribute HME Parameters
Key Distribution
Authentication request
Authentication request
2
2
3
Remote Attestation
Remote Attestation
3
4
Authenticate Researcher
Authenticate DOs
4
5
Generate HME Key Pairs (k^1_{pub}, k^1_{priv})
Distribute Public Key k^1_{pub}
Distribute Public Key k^1_{pub}
Distribute Public Key k^1_{pub}
5
Data Provisioning
Encrypt Training Data using k^1_{pub} (and CRT batching)
6
Upload Encrypted Training Data
SecureLR Model Training
Generate One-Time-Use Random Values → Encrypt using k^1_{pub}
Send Encrypted Random Values
Perform Secure Logistic Regression Computation under HME
7
Transfer Intermediate Gradient Values
8
1. Decrypt intermediate values
2. Perform Secure Scale-Down, Inner-Slot Aggregation of values
Model Retrieval
Generate HME Key Pairs (k^2_{pub}, k^2_{priv})
9
Distribute Public Key k^2_{pub}
1. Decrypt Trained Model Parameters within Enclave
2. Re-encrypt using k^2_{pub}
Send Back Encrypted Result
Decrypt Trained Parameters using k^2_{priv}

Fig. 2. SecureLR Protocol Workflow. Overview of our proposed SecureLR framework, consisting of five principal major stages: HME initialization, remote attestation and key distribution, data provisioning, SecureLR model training and model retrieving.

3.2 Remote Attestation and Key Distribution

In SGX [28], [30], the remote attestation process ensures that the code executed in the enclave is trusted, and a secure channel between the enclave and the enclave creator (i.e. ASP) has been established.

Step 2-4: Authentication Request (ARs, DOs→ASP) and Remote Attestation (ASP→CSP1, CSP2). The attestation process consists of several message exchanges between the enclave and the enclave creator, and may be viewed as a modified SIGMA protocol [41]. First, the public key of the client is sealed in the enclave, while the public key of the enclave is stored as part of the Intel Attestation Service (IAS). IAS is a part of ASP which provided directly by Intel and it is used to validate the authenticity and integrity of secure hardware environment. Therefore, to verify the authenticity of the enclave's signature, the client must pass it to the IAS for evaluation. The crypto-system for attestation of such keys is performed through an Enhanced Privacy ID algorithm, where a common group public verification key corresponds to a group of private keys. With the enclave's private key residing among this group, its uniqueness cannot be recovered via the public key, thus protecting its identity. The IAS then sends the response to the client to indicate whether the attestation process passes or not, and a secure channel is established for further communication.

Step 5: Generate and Distribute HME Key Pairs (CSP2→CSP1, DOs, ARs) After successful attestation, the verified enclave on CSP2 generates a pair of HME keys, the public key, k^1_{pub} and its associated private key, k^1_{sec}.

The k_{pub}^{1} will be distributed to DOs to encrypt their datasets for secure data outsourcing, as well as the ARs for encrypted random number generation to assist in secure computation between CSPs. We assume the security of private key k_{sec}^{1} is ensured by the secure enclave under the non-colluding semi-honest security model. In addition, the AR will generate another HME key pair k_{pub}^{2} and k_{sec}^{2}, where the public key k_{pub}^{2} will be sent to CSP2 to encrypt the resulting trained (regression) parameters. The details of encrypting these results are described in §3.4.

3.3 Data Provisioning and Encryption

Step 6: Encrypt and Upload Training Data (DOs→CSP1). The initial task after the verification of trusted parties involved in the proposed SecureLR protocol requires that the DOs encrypt all entries in their dataset (i.e. X and Y) using k_{pub}^{1} obtained from the verified enclave (hosted by the CSP2). A straightforward, *naive* method for encrypting this dataset is for DOs to individually encrypt the elements in X, or the x_i^j's, each in their respective ciphertexts; this results in $n * m$ ciphertexts and a significant increase in storage and computational overheads. As we mentioned in §3.1, the HME CRT-batching scheme (based on the Chinese Remainder Theorem [42]) allows for applying the same arithmetic operations to multiple elements in an SIMD manner, such that the slots within the operand ciphertexts undergo the arithmetic, which greatly reduces the space and computational complexity of the naive encryption approach. Intuitively, a CRT-batch can be thought of as having a structure similar to an array or column vector of encrypted values, where each value is stored in a slot, however instead of being accessible, the entire structure is encapsulated within a single unit. Thus, we may encrypt an entire dataset X comprised of m features into only m batched ciphertexts, where x^j, the column containing the j^{th} features of all n samples of X, becomes $\varepsilon(x^j) = \varepsilon([x_1^j, x_2^j, ..., x_n^j]^T)$, where ε denotes an encryption function of HME scheme . We utilize the same methodology to encrypt the observed class label Y into a single batched ciphertext, such that $\varepsilon(Y) = \varepsilon([y_1, y_2, ..., y_n]^T)$. A small caveat of using CRT-batching (in our implementation) requires the encrypted values be *unsigned integers*. To accommodate the *unsigned* requirement, meaning stored values must be positive, we represent negative values using Two's Complement number representation form. To address the need for *integer* values, we introduce a scaling factor β, with which to scale up all floating point numbers into integers with the precision of $1/\beta$. For example, given a number $a = 0.0123$ and a scaling factor $\beta = 1000$, the scaled integer will be $\hat{a} = truncate(a * \beta) = 12$, resulting in a one-digit loss in precision. The selection of β is based on the desired precision to be preserved in the fixed-point approximation of the floating numbers in the proposed framework. As the scaling factor is independent of the datasets, it is a fixed parameter, which could be chosen by the researcher before the secure computation and will not increase the privacy/security risk of the proposed framework. It is worth mentioning that there is a trade-off between computational performance and fixed-point approximation accuracy. Through several experiments, we found β =1000 provides a balanced trade-off.

Step 7: Generate and Encrypt Random Values for SMC (ARs→CSP1). The ARs requesting SecureLR services initialize and encrypt one-time-use random values to be employed during the logistic regression portion of the SecureLR protocol. ARs initialize three lists of such random values, and apply the same encryption technique (see §3.4, Algorithm 1).

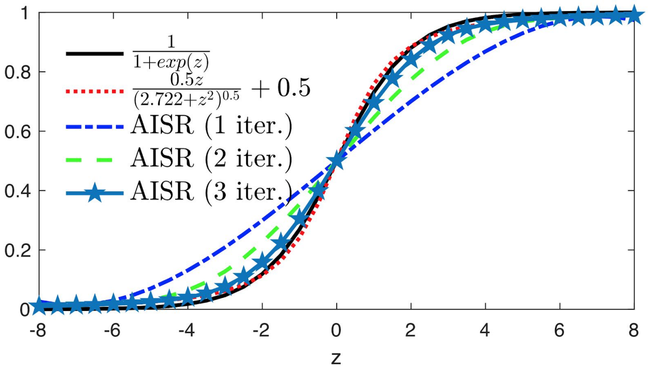


Fig. 3. Sigmoidal Function Comparison. Different methods of approximating the sigmoid function: an inverse of a square root (ISR) approximation, a new iterative approximation of ISR (AISR) using 1, 2 and 3 iterations of approximation on the input range of (-8,8).

3.4 SecureLR Model Training

We now discuss the problem of maximum likelihood estimation (§2.3 for an introduction). Though no closed-form solution exists, there are many algorithms that can be used to approximate its solution. Some of these involve variations on Newton's method and require matrix inversion, even a particularly efficient derivative known as Iteratively Reweighted Least Squares (IRLS), and others, such as the Fixed-Hessian Newton method, additionally require the inversion of a Hessian matrix, which may be especially costly under HME computation [43]. Although secure matrix inversion using a garbled circuit-based method [19] has been attempted, the scalability issue when using large datasets, and the requirement of synchronization among cooperating parties prevents us from implementing this in our protocol. With these considerations, we turn to another widely implemented algorithm we modify to support HME and SMC techniques we now discuss.

Step 8: Secure Logistic Regression under HME (CSP1, CSP2, ARs). To achieve maximum likelihood estimation, we take advantage of the *gradient ascent* algorithm to maximize the log-likelihood function (§2.3). This algorithm has been made famous by the resurgence in popularity of neural networks, and is more commonly seen referred to as gradient *descent*. We note that these are in essence identical algorithms, however in *ascent* we seek to find a local maximum, whereas the *descent* counterpart seeks to minimize a cost function, or minimize error by finding a local minimum. One benefit of gradient ascent over Newtonian methods is that it allows us to approximate the gradient of the log-likelihood function with respect to all training samples in each of its iterations, which we will herein refer to as *epochs*, and similarly perform an update to the regression model parameters without involving expensive matrix inversion (in the sense of encrypted operations). This allows us to develop an algorithm which utilizes SIMD operations on CRT-batched ciphertexts (see Algorithm 1).

In our SecureLR implementation, the model parameters are iteratively updated via gradient ascent, given by

$$w_j \leftarrow w_j + \alpha(y_i - g(w_j x_i))x_i^j$$

where $g(\cdot)$ is the sigmoid function, α is the learning rate, and w_j is the j^{th} parameter to update during each epoch with respect to the i^{th} sample. The remaining challenge is to transform this process to satisfy HME conditions without compromising efficiency or accuracy. As we cannot evaluate the exponential function nor division under HME, we must approximate the sigmoid function used in this algorithm.

A straightforward approximation of the sigmoid function using Taylor series introduces a substantial cost and has limited accuracy [21]. In the proposed SecureLR framework, we employ a novel approach to achieve this using alternative approximations with improved efficiency without sacrificing the performance of logistic regression. We use the approximation function $g_1(z) = \frac{0.5z}{\sqrt{2.722 + z^2}} + 0.5$ (see red dots line in Fig. 3) based on original formula given as $g(z) = \frac{az}{\sqrt{b+z^2}}$. We performed grid search on the hyperparameters a and b such that we can minimize the mean squared errors (MSE) between the approximation function the and original sigmoid function with in the range (-8,8), which closely mimics the sigmoid function (black line in Fig. 3). Both approximations rely on a key function that is the inverse of a square root, which fortunately, can be calculated efficiently using Newton's algorithm (only using addition and multiplication) [44]. More specifically, one can iteratively approximate this key function as $u_{k+1} = 0.5 \cdot u_k \cdot (3 - (2.722 + z^2) \cdot u_k^2)$, where k is the k^{th} iteration in this approximation of ISR (AISR) method. For example, using $u_0 = 0.14$ as an initial value, the blue star-dot line in Figure 3 shows the closeness in approximation after only 3 iterations of AISR.

We summarize the details of the proposed SecureLR framework in Algorithms 1 to 4. In Algorithm 1, lines 1-4, the DOs performs a one-time encryption of the local dataset using an efficient batching scheme to allow SIMD operation over $O(m)$ ciphertexts instead of $O(nm)$ ciphertexts as with the naive approach, vastly improving storage and secure computation performance. In lines 5-6, ARs initialize a SecureLR request to CSP with several parameters.

Lines 8-15 describe the proposed gradient ascent (GA) based SecureLR training algorithm. In line 9, we perform batched matrix-vector multiplication over homomorphic encrypted data resulting in $O(m)$ multiplication and addition operations compared with $O(nm^2)$ operations with naive dataset encryption. In line 10, we implement the proposed AISR method for efficiently approximating the logistic function used in GA.

As shown in Algorithm 2, we propose our design of a unique combination of security protocols through HME, where we perform the bulk of our secure computation over the encrypted dataset, and delegate the task of refreshing the accumulated ciphertext noise and rescaling of the underlying plaintext value (after a certain number of arithmetic operations under HME) to the enclave.

Algorithm 3 depicts the steps in line 14 of Algorithm 1, which we implement using the enclave in CSP2 to aggregate the inner-slots of batched ciphertexts due to a computational limitation of our current HME implementation, we also resort to CSP2 for the inner-slots aggregation of the batched ciphertexts to the enclave. To safely perform the aforementioned operations, one-time random values must first be added to such ciphertexts to mask out the original plaintexts before being sent to the enclave on CSP2. Since k_{pub}^{1} is shared with the ARs, they will initialize and encrypt these random values, then send them to CSP1 where the values are added to the intermediate ciphertexts as a form of mask, and the newly scrambled encrypted data can be safely transferred to CSP2 to undergo the operations.

Step 9: Re-encrypt Model Parameters (CSP2→ARs). In Algorithm 1, Lines 16 to 20 depict the secure model parameter retrieval protocol, where the garbled model parameters are re-encrypted in the enclave with the k^2_{pub} (described in Algorithm 4). Upon receiving the securely encoded model parameters, ARs are able to remove their self-generated random values and recover the original, trained parameters.

Algorithm 1: SecureLR using Gradient Ascent and a Hybrid, Secure Model

Notation: $\varepsilon(z) \leftarrow z$: batched encryption of a plaintext vector z with key^1_{pub} (where each slot of $\varepsilon(z)$ stores an element of vector z)
$\varepsilon'(z)$ denotes the ciphertext encrypted by key^2_{pub} (see §3.3).
β : public *scaling factor* to encode floating-point numbers as integers.
i, j, k_R, k_S, l :the i-th slot in a ciphertext, the indices of the j-th feature in X, the k_R-th random noise in $\varepsilon(R)$, the k_S-th random noise in $\varepsilon(S)$, the l-th iteration in GA, respectively.
k++: post-increment operator which increases k after the current value of k is used.

Lines: Given proper remote attestation, authorization, and dissemination of encryption keys as depicted in Figure 2.

```
    Note that unless specifically stated, all operations are under HME on CSP1.
Using $key^{1}_{pub}$ from the authenticated enclave, DO performs a one-time encryption of the dataset (locally):
    Encryption of X: for-each $j = 0$ to $m-1$: $\varepsilon(x^j) \leftarrow x^j$ (where the bias term $x^m$ is a known value of 1's and not encrypted to optimize the number of HME operations)
    Encryption of scaled Y: $\varepsilon(Y_\beta) \leftarrow Y * \beta$
    DOs send encrypted dataset to CSP1
AR initializes a SecureLR request to CSP1 with the parameters as follows:
    Initialization of model parameters in plaintext:
    $\alpha$ (learning rate), $N$ (# of epochs in GA), $L$ (# of iterations in AISR), $u_0$ (initial value for AISR),
    $\varepsilon(R) = \{\varepsilon(r_1), \varepsilon(r^{s}_{1}), \varepsilon(r_2), \varepsilon(r^{s}_{2}), ..., \varepsilon(r_K), \varepsilon(r^{s}_{K_R})\}$ - a list of one-time-use encrypted random values for secure scaling operations within the enclave, where $K_R = N * L$ and $r^{s}_{k} = Round(r_k/\beta)$
    $\varepsilon(S) = \{\varepsilon(s_1), \varepsilon(s^{a}_{1}), \varepsilon(s_2), \varepsilon(s^{a}_{2}), ..., \varepsilon(s_K), \varepsilon(s^{a}_{K_S})\}$ - a list of one-time-use encrypted random values for secure inner-slot aggregation within the enclave, where $K_S = N * m$ and $s^{a}_{i} = \sum_{i=1}^{n} s^{i}_{k}$
    $\varepsilon(O) = \{\varepsilon(o_1), \varepsilon(o_2), ..., \varepsilon(o_m)\}$ - a list of one-time-use encrypted random values for secure re-encryption operations within the enclave
    Initialization and encryption of scaled model (regression) parameters (values randomly generated between 0, 1):
    for $j = 0$ to $m$:
        $\varepsilon(w^j) \leftarrow rand(0, 1) * \beta$
    end-for
Gradient Ascent (GA) based:
    Set $k_S = 0$ and $k_R = 0$
    for-each epoch $l = 1$ to $N$ in GA
        Securely evaluate $v = X \cdot w$ efficiently using CRT-batching:
            Initialization of $\varepsilon(v)$ as $\varepsilon(v) = \varepsilon(w^m)$
            for $j = 1$ to $m-1$:
                $\varepsilon(v) \leftarrow \varepsilon(v) + \varepsilon(x^j) * \varepsilon(w^j)$
            end-for
        Secure approximate evaluation of $v_s = AISR(v) \sim g(v) = 1/(1 + exp(-v))$ using AISR with output of $\varepsilon(v_s)$ on CSP2 (see Algorithm 2)
        Compute $\varepsilon(e^L) = \varepsilon(v_s) - \varepsilon(y)$
        for $j = 1$ to $m$:
            if j < m: $\varepsilon(e^j) = \varepsilon(e^L) * \varepsilon(x^j)$
            Secure inner-slot aggregation of $\varepsilon(a^j) = \varepsilon(\sum_{i=1}^{n} e^{j}_{i})$ on CSP2 (see Algorithm 3)
            Securely update model parameters: $\varepsilon(w^j) \leftarrow \varepsilon(w^j) + \alpha * \varepsilon(a^j)$
        end-for
    end-for-each
Securely retrieve trained model (regression) parameters:
    for $j = 1$ to $m$:
        Secure re-encryption of $\varepsilon(w^j)$ using $key^{2}_{pub}$ with output of $\varepsilon'(w^j)$ on CSP2 (see Algorithm 4)
        CSP 1 sends $\varepsilon'(w^j)$ back to AR for model parameter decryption
        AR computes $\varepsilon'(w^j) = \varepsilon'(w^j) - o_j$ and performs decryption with $key^{2}_{sec}$ to obtain the final parameter $w^j$
    end-for
```

Algorithm 2: Secure AISR Evaluation

```
Lines:  Given $u_0$, $\varepsilon(R)$, $k_R$, $v$ in Algorithm 1
for i = 1 to L
    $\varepsilon(u_i) = 0.5 \cdot \varepsilon(u_{i-1}) \cdot (3 - (2.722 + \varepsilon(v)^2) \cdot \varepsilon(u_{i-1})^2)$
    Securely rescale $\varepsilon(u_i)$:
        CSP1 sends $\varepsilon(u'_i) \leftarrow \varepsilon(u_i) + \varepsilon(r_{k_R})$ to the secure enclave hosted by CSP2
        Enclave decrypts $\varepsilon(u'_i)$ followed by a scaling-down operation with a factor of $\beta$ (i.e., $u''_i = u'_i/\beta$);
        Return re-encrypted $\varepsilon(u''_i)$ to CSP1 to remove the additive noise as
         $\varepsilon(u_i) \leftarrow \varepsilon(u''_i) - \varepsilon(r^{s}_{k_R++})$
Return: $\varepsilon(u_L)$
```

Algorithm 3: Secure Aggregation of Ciphertext Inner-Slots

```
Lines:  Given $\varepsilon(e^j)$, $\varepsilon(S)$, $k_S$, $v$ in Algorithm 1
CSP 1 sends $\varepsilon(e^j) = \varepsilon(e^j) + \varepsilon(s_{k_S})$ to the secure enclave hosted by CSP 2
```

```
  *Enclave* decrypts $\varepsilon(e^j)$ followed by an inner-slot aggregation computation as $a^j = \sum_{i=1}^{n} e^j{}_i$
  Return re-encrypted $\varepsilon(a^j)$ to CSP 1 to remove the additive noise as
  $\varepsilon(a^j) \leftarrow \varepsilon(a^j) - \varepsilon(s^a{}_{k_S++})$
Return: $\varepsilon(a^j)$
```

Algorithm 4: Secure Re-encryption of Model Parameters

```
Lines: Given $j$, $\varepsilon(w^j)$, $\varepsilon(O)$, $key^2{}_{pub}$ in Algorithm 1
  *CSP1* sends $\varepsilon(w^j) = \varepsilon(w^j) + (\varepsilon(o_j))$ to the *enclave* hosted by *CSP2*
      *Enclave* decrypts $\varepsilon(w^j)$, then re-encrypts with $key^2{}_{pub}$: $\varepsilon'(w^j) \leftarrow w^j$
Return: $\varepsilon'(w^j)$
```

4 Experimental Setup

We implement the proposed SecureLR framework in C++ using the Microsoft SEAL v2.1 HME library [45] and Intel® SGX, SDK v1.7. We evaluate all experiments on an SGX-enabled PC with a 2.7GHz Intel® Core™ i7-6820HK (x64 based) processor and 48.0GB RAM (Windows 10 Pro), connected via Ethernet. For homomorphic encryption, we select the following parameters to achieve sufficient security and correctness for use of NTT and CRT batching: the polynomial modulus $p = 1 * x^{8192} + 1$, $n = 8192$, security parameter $\lambda = 80$, coefficient modulus $q = 2^{226} - 2^{26} + 1$, and plaintext modulus $t = 1073692673$. These parameters satisfy

$$n \geq \frac{\lambda + 110}{7.2} * log(q)$$

and result in batched ciphertexts with a capacity of 8192 slots, such that each ciphertext uses 512.0KB of memory [46]. Logistic regression parameters require tuning, and are typically selected via cross-validation over plaintext. Given the nature of the SecureLR, manually tuning the model's hyperparameters may pose as a challenge under encryption for users (ARs), though given the computational power of the CSP, applying a Grid Search approach to hyperparameter optimization may be feasible (e.g. with cooperation between the CSPs and ARs to implement an early stopping technique to know when to halt the training process when a poor combination is being tested, or test the model to know when to select a near-optimal one). Thus for our demonstration, we show results for both model and dataset-specific, tuned hyperparameters, as well as by fixing α, N, and u_0 as α = 0.01, N = 20, and u_0 = 0.001 (Table 2). Experimental results are obtained through 10-fold cross-validation.

5 Experimental Results

We compare our proposed SecureLR framework with logistic regression over the plaintext datasets to measure the model's accuracy (in terms of the AUC, or Area Under the Curve), space complexity to demonstrate approximated storage costs, and runtime of model training as our performance metric. The size of a single batched ciphertext is based on the aforementioned HME parameters we select for our experiments to accommodate all datasets we test on. We summarize the datasets we use for all evaluative experiments in Table 1 (the first three are biomedical, and the last from an original comparison on machine learning algorithms).

TABLE 1
Description of Datasets

Datasets	Edinburgh MI [47]	SPECT Heart [48]	WI-Breast Cancer [49]	MONK's Prob. [50]
# of samples	1252	267	699	556
# of features	10	23	10	7
Ciphertext (MB)	5.01	11.5	5.01	3.51
Plaintext (KB)	34.3	19.6	25.2	14.7

Details of the four datasets used in our experiments. Ciphertext and plaintext indicate the size before and after batch-encryption

The overall space complexity of our protocol depends principally upon the size of the dataset; in particular, the number of features or attributes (i.e. m) indicates the base size of encrypted data in SecureLR, as we demonstrate for our experimental datasets in Table 1. The selection of the number of epochs of gradient ascent and number of iterations of the AISR algorithm (i.e. N and L, respectively) determines the additional cost of the one-time random number generation by the ARs used in Algorithms 2 and 3 for security against advanced attacks within *CSP1* and *CSP2*. Hence, the overall space complexity of SecureLR is given by $O(m) + O(N * L) + O(N)$.

We choose to demonstrate the accuracy of SecureLR by displaying our results for two distinct tests (given in Table 2). The first test which shows the comparative accuracy of SecureLR using AISR-1 and AISR-3 against logistic regression over the plaintext datasets using the standard sigmoid function, $g_1(z) = 1/1 + exp(-z)$ and the unapproximated ISR function $g_2(z) = \frac{0.5z}{\sqrt{2.722 + z^2}} + 0.5$. We denote these as *trivial*, as we fixed the logistic regression hyperparameters (α, N, and u_0), not necessarily optimized for any particular set of functions and datasets, and evaluated all datasets and models using these.

TABLE 2
Accuracy of Different Models

Datasets	AUC					
	Trivial (Fixed HP)				Optimized	
	Sigmoid	ISR	AISR-1	AISR-3	AISR-1*	AISR-3*
Edin.	0.972	0.950	0.595	0.623	0.941	0.956
SPECT	0.842	0.854	0.762	0.797	0.849	0.886
WIBC	0.955	0.946	0.889	0.875	0.960	0.980
MONK's	0.741	0.727	0.742	0.740	0.766	0.776

Comparison of AUC performance between SecureLR using AISR and logistic regression over plaintext using Sigmoid and ISR.

TABLE 3
HYPERPARAMETER SELECTION USED IN TABLE 2

Datasets	Fixed HP			AISR-1*			AISR-3*		
	u_0	α	N	u_0	α	N	u_0	α	N
Edin.	0.001	0.01	20	0.01	0.06	5	0.01	0.07	5
SPECT				0.01	0.008	10	0.01	0.008	5
WIBC				0.01	0.002	10	0.01	0.003	10
MONK's				0.01	0.005	5	0.001	0.03	10

Description of hyperparameters used to demonstrate accuracy of the SecureLR protocol.

The nature of hyperparameter selection is well-known to be dataset or problem dependent in the machine learning community, and to address this potential difficult, we show the performance of SecureLR with hyperparameters tuned for each dataset and AISR-1* and AISR-3* using a modified Grid Search approach, as well as the results for hyperparameters selected to somewhat suit all four models and datasets (an often challenging task). We do this to demonstrate the protocol's performance in the case that a potential user (ARs) chooses to utilize the computing power of the CSP to perform hyper parameter optimization on their dataset, or select "somewhat default" settings. Empirically, we find that the "somewhat default" settings may require more epochs to achieve convergence (i.e. in the case of the Edinburgh MI dataset for AISR-1 and AISR-3, we achieve a high AUC$\geq 0.90 - 0.96$ for both approximations after 200 epochs.

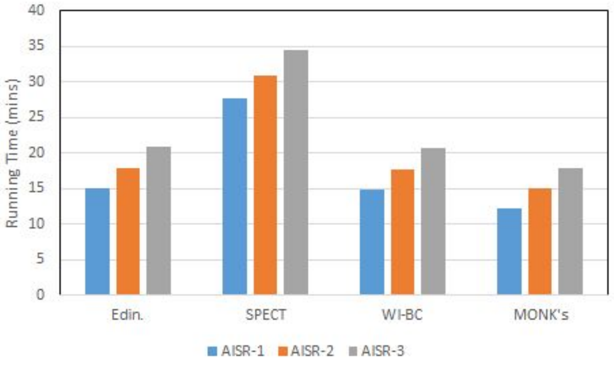


Fig. 4. Runtime of Aisr-1, 2 & 3. Overall runtime of SecureLR using 1, 2, and 3 iterations of the AISR approximation algorithms for each dataset (in minutes).

The proposed SecureLR achieves comparable AUC performance for most testing datasets. However, it is worth mentioning that there is a slower rate of convergence for the Edinburgh MI dataset under the default parameter settings. In Table 2, we demonstrated that we can improve the AUC performance by increasing either α or u_0 by a factor of 10 to improve the rate of convergence when this occurs. We admit that this is one of limitations of gradient ascent based optimization methods, where it may need fine-tuned hyper-parameter settings to achieve the best AUC performance for some datasets.

We demonstrate the runtime of SecureLR corresponding to the results for AISR-1 and AISR-3 with 20 epochs of gradient ascent, depicting the total training time (Fig. 4) and average time per epoch, corresponding to the experiments using fixed hyperparameters. Our experiments demonstrate that for some combinations of α, N, and u_0, there may be an improvement in accuracy using L=3 versus $L=1$ in the AISR algorithm with a trade-off of a small increase in runtime per epoch. Empirically, the cost of using more iterations increases linearly with respect to the size of the input.

TABLE 4
AVERAGE TIME PER EPOCH OF AISR-1, 2 & 3

	Time per epoch (s)		
Datasets	**AISR-1**	**AISR-2**	**AISR-3**
Edin.	44.9	53.6	62.4
SPECT	82.8	92.4	102
WI-BC	44.6	53.2	61.8
MONK's	36.0	44.6	53.3

Average runtime per epoch of SecureLR using 1, 2, and 3 iterations of the AISR approximation algorithms (in seconds).

To further demonstrate the performance of SecureLR, we compare it with two existing SMC-based secure logistic regression protocols. Firstly, the privacy-preserving model of *Aono et al.* [51] employs a method which makes use of secure two-party computation. The space complexity of their model is $O(nm^2)$, where n denotes the number of training samples and m the number of features), whereas, the base space complexity of our protocol is given by $O(m)$ to encode the given dataset, plus $O(N*L)+O(N)$ additional ciphertexts used in the secure logistic regression computation. Similarly, the model by Aono et al. demonstrates a time complexity of $O(nm^2)$. Our protocol uses $O(m*N*L)$ time for secure logistic regression with N epochs for gradient ascent, wherein each requires m steps to compute the product of Xw with L iterations of AISR approximation.

Next, we examine the tradeoffs between the SMAC-GLORE model by Wang et al. [19], which employs secure multi-party computation to build a Grid logistic regression model. As demonstrated by their experiments, the model requires 47 minutes, to compute logistic regression training over a small dataset containing 60 samples with 3 binary-encoded features. In comparison, we train our model on a dataset containing 699 samples and 10 attributes in less than 10 and a half minutes (on average for a full instance of SecureLR training). Thus, our protocol offers both computational efficiency and accuracy with real world, practical applications.

6 DISCUSSION

In the proposed SecureLR framework, there are two secrets which are subjected to restricted privacy policy: the private biomedical data from DOs and the trained model results for ARs. Since throughout the process, ASP only handles attestation and authentication tasks, we consider that ASP will not have access to the aforementioned encryption keys. Thus, we concentrate on the following potential information leakage situations to demonstrate the SecureLR model is secure and privacy-preserving.

1. **DOs→CSPs**: All data sent from the DOs to CSP1 is protected under HME. The pairs of encryption keys used in this process are generated within an authenticated enclave on CSP2, which is sealed off from unauthorized access. Given our assumption that the CSPs are non-colluding and semi-honest parties, and all data remains in encrypted form while on CSP1, we maintain our data security guarantee within this scope. Additionally, all data sent to CSP2 are masked with one-time-use random values prior to decryption inside its enclave. Thus these data remain private and protected during this process.
2. **DOs→ARs**: The ARs receive trained model parameters in the final step of the SecureLR protocol, which are encrypted and decrypted using $k^2{}_{pub}$ and $k^2{}_{sec}$ respectively. This separate pair of keys are specific to the ARs, and are unable to decrypt the DOs' dataset. ARs are furthermore unable to learn information regarding the plaintext dataset used to train these parameters.
3. **ARs→CSPs**: During the training process, all the intermediate results are either encrypted under HME on CSP1 or masked by random values before being decrypted in the enclave on CSP2. The model training results remain encrypted and masked by random values while stored on CSPs. Thus, CSPs do not have the ability to access the training data nor resulting model parameters.

The proposed SecureLR framework has a few limitations. To use computationally efficient HME techniques under our current design, including CRT-batch encryption and moderately sized ciphertexts, our implementation requires that we store encrypted values as integers. To represent floating-point numbers in the correct form, we must first apply a scale factor to encode them up to a specified level of precision. Increasing this level of precision without compromising encryption security results in a larger ciphertext, which in turn leads to increases in both the computational and storage costs of the model. In addition, we introduced an algorithmic method to approximate the sigmoid function (AISR); while this algorithm evaluates a simpler circuit than the standard logistic function in which the exponential function must be computed, the approximation accuracy of this approach depends upon the initial value, u_0 and the number of iterations, L. The resulting loss in precision may affect the model's rate of convergence or accuracy. The proposed SecureLR framework uses Gradient Ascent, which requires more iterations to tune the regression parameters to maximize the likelihood estimation than the Newton-Raphson algorithm and other Newtonian methods.

In employing SGX, we assume that the secret key it stores will not be stolen by a malicious party or attackers. Besides this, our solution is immune to controlled-channel and cache-timing attacks, as SGX has recently been shown with vulnerabilities to those attacks [52]. As we only let SGX do very limited operations and utilize secure multi-party computation techniques when sharing encrypted data with it, our solution has significantly reduced the attack surface within the enclave.

7 Conclusion

In this paper, we propose a framework for conducting privacy-preserving logistic regression based on a hybrid cryptographic model using homomorphic encryption and secure hardware, ideal for learning from biomedical and sensitive data in a public cloud environment. Our proposed method provides a multi-faceted data provisioning approach: we take advantage of homomorphic CRT-batching methods for increased computational efficacy through SIMD operations and reduced storage demands, as well as incorporate an SMC-inspired techniques to mask out the underlying plaintext values between two CSPs. Our method allows for practical use of logistic regression on health and other patient-linked data in the public cloud. In the future, we will extend our protocol to support secure and efficient multinomial/multiclass logistic regression and other biomedical data analysis methods.

Acknowledgment

This work is supported by the R00HG008175, R01GM114612, U01EB023685, R21HG009961, NSERC Discovery Grants (RGPIN-2015-04147) and University of Manitoba Startup Grant.

References

[1] S.-A. Sansone *et al.*, "DATS: the data tag suite to enable discoverability of datasets," *bioRxiv*, p. 103143, 25-Jan-2017.

[2] "All of Us Research Program," *National Institutes of Health (NIH)*. [Online]. Available: https://www.nih.gov/research-training/allofus-research-program. [Accessed: 07-Feb-2017].

[3] "Precision Medicine Initiative | National Institutes of Health (NIH)." .

[4] S. of California, "Investigation of major Anthem cyber breach reveals foreign nation behind breach." [Online]. Available: http://www.insurance.ca.gov/0400-news/0100-press-releases/2017/release001-17.cfm. [Accessed: 30-Jul-2017].

[5] M. Naveed *et al.*, "Privacy in the Genomic Era," *ACM Comput Surv*, vol. 48, no. 1, Sep. 2015.

[6] B. Malin, D. Karp, and R. H. Scheuermann, "Technical and policy approaches to balancing patient privacy and data sharing in clinical and translational research," *J. Investig. Med.*, vol. 58, no. 1, pp. 11–18, Jan. 2010.

[7] S. N. Murphy, V. Gainer, M. Mendis, S. Churchill, and I. Kohane, "Strategies for maintaining patient privacy in i2b2," *J. Am. Med. Inform. Assoc.*, vol. 18 Suppl 1, pp. i103–8, Dec. 2011.

[8] H. Tang *et al.*, "Protecting genomic data analytics in the cloud: state of the art and opportunities," *BMC Med. Genomics*, vol. 9, no. 1, p. 63, Oct. 2016.

[9] "Health Insurance Portability and Accountability Act (HIPAA)." .

[10] HealthITSecurity, "How are Healthcare Data Breach Victims Affected by Attacks?," *HealthITSecurity*, 13-Sep-2016. [Online]. Available: https://healthitsecurity.com/news/how-are-healthcare-data-breach-victims-affected-by-attacks. [Accessed: 01-Sep-2017].

[11] E. C. McKiernan *et al.*, "How open science helps researchers succeed," *Elife*, vol. 5, Jul. 2016.

[12] P. S. Kamath and W. Kim, "The model for end-stage liver disease (MELD)," *Hepatology*, vol. 45, no. 3, pp. 797–805, 2007.

[13] R. L. Kennedy, A. M. Burton, H. S. Fraser, L. N. McStay, and R. F. Harrison, "Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: derivation and evaluation of logistic regression models," *Eur. Heart J.*, vol. 17, no. 8, pp. 1181–1191, Aug. 1996.

[14] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in Neural Information Processing Systems*, 2008, pp. 289–296.

[15] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional Mechanism: Regression Analysis Under Differential Privacy," *Proceedings VLDB Endowment*, vol. 5, no. 11, pp. 1364–1375, Jul. 2012.

[16] Y. Wu, X. Jiang, J. Kim, and L. Ohno-Machado, "Grid Binary LOgistic REgression (GLORE): building shared models without sharing data.," *J. Am. Med. Inform. Assoc.*, vol. 2012, no. 5, pp. 758–764, Apr. 2012.

[17] S. Wang, X. Jiang, Y. Wu, L. Cui, S. Cheng, and L. Ohno-Machado, "EXpectation Propagation LOgistic REgRession (EXPLORER): distributed privacy-preserving online model learning," *J. Biomed. Inform.*, vol. 46, no. 3, pp. 480–496, Jun. 2013.

[18] Y. Wu, X. Jiang, S. Wang, W. Jiang, P. Li, and L. Ohno-Machado, "Grid multi-category response logistic models," *BMC Med. Inform. Decis. Mak.*, vol. (15), no. 1, pp. 1–10, 2015.

[19] H. Shi *et al.*, "Secure Multi-pArty Computation Grid LOgistic REgression (SMAC-GLORE)," *BMC Med. Inform. Decis. Mak.*, vol. 16 Suppl 3, p. 89, Jul. 2016.

[20] W. Xie, Y. Wang, S. M. Boker, and D. E. Brown, "PrivLogit: Efficient Privacy-preserving Logistic Regression by Tailoring Numerical Optimizers," *arXiv [cs.LG]*, 03-Nov-2016.

[21] Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang, "Scalable and Secure Logistic Regression via Homomorphic Encryption," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, New Orleans, Louisiana, USA, 2016, pp. 142–144.

[22] K. El Emam, S. Samet, L. Arbuckle, R. Tamblyn, C. Earle, and M. Kantarcioglu, "A secure distributed logistic regression protocol for the detection of rare adverse drug events," *J. Am. Med. Inform. Assoc.*, vol. 20, no. 3, pp. 453–461, 2013.

[23] M. Kim and K. Lauter, "Private genome analysis through homomorphic encryption," *BMC Med. Inform. Decis. Mak.*, vol. 15 Suppl 5, p. S3, Dec. 2015.

[24] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can Homomorphic Encryption Be Practical?," in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, Chicago, Illinois, USA, 2011, pp. 113–124.

[25] F. Chen *et al.*, "PRINCESS: Privacy-protecting Rare disease International Network Collaboration via Encryption through Software guard extensionS," *Bioinformatics*, vol. 33, no. 6, pp. 871–878, 2017.

[26] F. Chen, M. Dow, S. Ding, and Others, "PREMIX: PRivacy-preserving EstiMation of Individual admiXture," in *American Medical Informatics Association Annual Symposium*, Chicago, 2016.

[27] F. Chen *et al.*, "PRESAGE: PRivacy-preserving gEnetic testing via SoftwAre Guard Extension," *BMC Medical Genomics*, vol. 10, no. 2, p. 48, 2017.

[28] V. Costan and S. Devadas, "Intel sgx explained," Cryptology ePrint Archive, Report 2016/086, 2016. https://eprint. iacr. org/2016/086.

[29] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Security and Privacy (SP), 2015 IEEE Symposium on*, 2015, pp. 640–656.

[30] "Intel® Software Guard Extensions (Intel® SGX)." [Online]. Available: https://software.intel.com/en-us/isa-extensions/intel-sgx. [Accessed: 01-Feb-2016].

[31] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, 1982, pp. 160–164.

[32] F. Armknecht *et al.*, "A Guide to Fully Homomorphic Encryption."

[33] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[34] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," in *Advances in Cryptology – EUROCRYPT 2010*, 2010, pp. 1–23.

[35] admin, "Intel SGX Homepage | Intel® Software," *Intel*, 28-Mar-2016. [Online]. Available: http://software.intel.com/en-us/sgx. [Accessed: 15-Aug-2017].

[36] G. Rodríguez, "Lectures notes about generalized linear models," 2008.

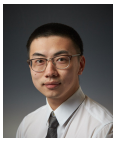
[37] Z. Huang, H. Lin, J. Fellay, Z. Kutalik, and J.-P. Hubaux, "SQC: secure quality control for meta-analysis of genome-wide association studies," *Bioinformatics*, vol. 33, no. 15, pp. 2273–2280, Aug. 2017.

[38] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, and G. Bejerano, "Deriving genomic diagnoses without revealing patient genomes," *Science*, vol. 357, no. 6352, pp. 692–695, Aug. 2017.

[39] S. Halevi and V. Shoup, "Algorithms in HElib," *Advances in Cryptology--CRYPTO 2014*, pp. 554–571, 2014.

[40] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic

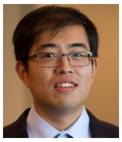
encryption for arithmetic of approximate numbers," IACR Cryptology ePrint Archive, 2016: 421, 2016.
[41] admin, "Intel® Software Guard Extensions Remote Attestation End-to-End Example | Intel® Software," *Intel*, 08-Jul-2016. [Online]. Available: https://software.intel.com/en-us/articles/intel-software-guard-extensions-remote-attestation-end-to-end-example. [Accessed: 30-Jul-2017].
[42] C. Ding, D. Pei, and A. Salomaa, *Chinese remainder theorem: applications in computing, coding, cryptography*. World Scientific, 1996.
[43] T. P. Minka, "A comparison of numerical optimizers for logistic regression," Oct. 2003.
[44] K. Turkowski, "Computing the inverse square root," *Graphics Gems V*, pp. 16–21, 1995.
[45] K. Laine and R. Player, "Simple Encrypted Arithmetic Library-SEAL (v2. 0)," Technical report, September, 2016.
[46] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," *Advances in Cryptology–CRYPTO 2012*, pp. 850–867, 2012.
[47] X. Jiang, M. Osl, J. Kim, and L. Ohno-Machado, "Calibrating predictive model estimates to support personalized medicine," *J. Am. Med. Inform. Assoc.*, vol. 19, no. 2, pp. 263–274, Mar. 2012.
[48] "UCI Machine Learning Repository: Data Sets." [Online]. Available: https://archive.ics.uci.edu/ml/datasets. [Accessed: 25-Aug-2017].
[49] "UCI Machine Learning Repository: Breast Cancer Wisconsin (Original) Data Set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28original%29. [Accessed: 25-Aug-2017].
[50] "The MONK's Problems: A Performance Comparison of Different Learning Algorithms." [Online]. Available: http://robots.stanford.edu/papers/thrun.MONK.html. [Accessed: 21-Aug-2017].
[51] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, "Privacy-preserving logistic regression with distributed data sources via homomorphic encryption," *IEICE Trans. Inf. Syst.*, vol. 99, no. 8, pp. 2079–2089, 2016.
[52] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware Guard Extension: Using SGX to Conceal Cache Attacks," *arXiv [cs.CR]*, 28-Feb-2017.



Yichen Jiang achieved Master degree of Computer Science from Syracuse University in 2017. Now he is employed by UCSD department of biomedical informatics, his research focuses on machine learning and secure computing.

Jenny Hamer is a fourth year undergraduate student in Mathematics and Computer Science at the University of California, San Diego (B.S. expected 2018) with Provost Honors. She is a research assistant and software engineer in the Dept. of Biomedical Informatics, UCSD, and volunteers with the Voytek Lab, UCSD, in theoretical and computational cognitive science. Her research interests include machine and deep learning, cognition and neural modeling, and bioinformatical analyses.

Chenghong Wang received his bachelor degree in information security from Harbin Engineering University and master of science degree in computer science from Syracuse University. His research interests include graph theory, theoretical machine learning, security and privacy. He is the student member of ACM and IEEE since 2015.



Xiaoqian Jiang (S'06–M'10) is an associate professor in the Department of Biomedical Informatics, UCSD. He received his PhD in Computer Science from Carnegie Mellon University. He is an associate editor of BMC Medical Informatics and Decision Making and serves as an editorial board member of Journal of American Medical Informatics Association. He works primarily in health data privacy and predictive models in biomedicine. Dr. Jiang won the distinguished paper award from AMIA Clinical Research Informatics (CRI) Summit in 2012 and 2013.

Miran Kim received a Ph.D degree in Mathematical Sciences from Seoul National University, Seoul, Korea, in 2017. Currently, she is a postdoctoral researcher in the Department of Biomedical Informatics, UCSD. Her research interests include cryptography, computational number theory, genome data privacy, and machine learning.

Yongsoo Song received a PH.D degree in Mathematical Sciences from Seoul National University, Seoul, Korea, in 2018. Currently he is a postdoctoral researcher in the Department of Computer Science and Engineering, UCSD. His research interests include cryptography primitives for secure computation and their applications.

Yuhou Xia received her Ph.D. degree in mathematics from Princeton University in 2018. Her research interests include algebraic number theory, cryptography, machine learning and data privacy and security.

Md Nazmus Sadat received his B.Sc. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) in 2014. He is currently pursuing M.Sc. in Computer Science at University of Manitoba. His primary research interests include trusted hardware assisted secure computation techniques, privacy-preserving data analytics, and genomic data privacy.

Noman Mohammed received a Ph.D. degree in Computer Science from Concordia University in 2012. He is currently an Assistant Professor in the Department of Computer Science at University of Manitoba, Manitoba, Canada. Before coming to UofM, he was an NSERC postdoctoral fellow in the School of Computer Science at McGill University and a member of the Cryptography, Security, & Privacy (CrySP) Research Group at the University of Waterloo. His research interests include private data sharing, privacy-preserving data mining, secure distributed systems, and applied cryptography.

Shuang Wang (S'08–M'12) received the B.S. degree in applied physics and the M.S. degree in biomedical engineering from the Dalian University of Technology, China, and the Ph.D. degree in electrical and computer engineering from the University of Oklahoma, OK, USA, in 2012. He was worked as a postdoc researcher with the Department of Biomedical Informatics (DBMI), University of California, San Diego (UCSD), CA, USA, 2012 - 2015. Currently, he is an assistant professor at the DBMI, UCSD. His research interests include machine learning, and healthcare data privacy/security. He has published more than 60 journal/conference papers, 1 book and 2 book chapters. He was awarded a NGHRI K99/R00 career grant. Dr. Wang is a senior member of IEEE. Dr. Wang is a guest editor of Plos Genetics.