HUMAN BIOLOGY
2024, VOL. 94, ISSUE 4
ORIGINAL ARTICLE

Secure Inference on Homomorphically Encrypted Genotype Data with Encrypted Linear Models

Meng Zou[1*], *Guangyang Zhang*[1#], *Fan Zhang*[2#], *Guoping Liu*[1*]

ABSTRACT

Background: Accurate models are crucial to estimate the phenotypes from high throughput genomic data. While the genetic and phenotypic data are sensitive, secure models are essential to protect the private information. Therefore, construct an accurate and secure model is significant in secure inference of phenotypes.

Methods: We propose a secure inference protocol on homomorphically encrypted genotype data with encrypted linear models. Firstly, scale the genotype data by feature importance with Xgboost or Adaboost then train linear models to predict the phenotypes in plaintext. Secondly, encrypt the model parameters and test data with CKKS scheme for secure inference. Thirdly, predict the phenotypes under CKKS homomorphically encryption computation. Finally, decrypt the encrypted predictions by client to compute the 1-NRMSE/AUC for model evaluation.

Results: 5 phenotypes of 3000 samples with 20390 variants are used to validate the performance of the secure inference protocol. The protocol achieves 0.9548, 0.9639, 0.9673 (1-NRMSE) for 3 continuous phenotypes and 0.9943, 0.99290 (AUC) for 2 category phenotypes in test data. Moreover, the protocol shows robust in 100 times of random sampling. Furthermore, the protocol achieves 0.9725 (the average accuracy) in an encrypted test set with 198 samples, and it only takes 4.32s for the overall inference. These help the protocol rank top one in the iDASH-2022 track2 challenge.

Conclusion: We propose an accurate and secure protocol to predict the phenotype from genotype and it takes seconds to obtain hundreds of predictions for all phenotypes.

INTRODUCTION

The research on genotype-to-phenotype is crucial to uncover the gene functions and the mechanisms in distinct phenotypic outcomes Benfer et al. (2008). The high throughput genomic data makes it may be possible to predict the phenotype from genotype. While the inference of genotype to phenotype is a complex problem due to the intricate factors such as genotypes, epigenetic variants and their interactions Dowell et al. (2010). Moreover, an individual of same genotype may develop to thousands of different diseases, which makes it still a huge challenge in achieving accurate predictions for these phenotypes efficiently Lehner et al. (2013). Because of the sensitive nature of genotype and phenotype data, a secure and accurate model is essential for the secure inference of the predictions.

Linear or logistic regression models are generally applied in Genome-wide association studies (GWASs) such as SNPTEST and PLINK Burton et al. (2007), Purcell et al. (2007), Marchini et al. (2007). However, the linear models may be overfitting because of the number of genotypes far exceed phenotypic outcomes.

Regularized linear regression models such as ridge regression, lasso, elastic net and their extensions could overcome the overfitting problems and select a functional genotype set for phenotype estimations Rakitsch et al. (2012), Waldmann et al. (2013), Ogutu et al. (2012). While linear models could only capture additive effects, ensemble-based machining learning methods such as Xgboost or Adaboost could select the epistasis genotypes as well and may achieve better performance Banerjee et al. (2020). Either linear or non-linear models assist to construct an accurate model to infer the phenotype from genotype.

The sensitive nature of genotype and phenotype data urges to develop the secure inference models for phenotype prediction. Besides, the track 2 of iDASH-2022 appeals to develop a secure model evaluation on homomorphic encrypted genotype data via protecting both model parameters and genotypes.

[1]School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China
[2]Tencent, Beijing 100193, China

Corresponding to: Dr. Meng Zou, School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: zoumeng@hust.edu.cn.

Dr. Guoping Liu, School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: liuguoping@hust.edu.cn.

Keywords : homomorphic encryption, genotype to phenotype, CKKS, iDASH- 2022.

Homomorphic encryption (HE) is a cryptosystem that enables homomorphic operations on encrypted data and is considered as one of the most important primitives for privacy-preserving applications. Most of the current HE schemes can be categorized into word-wise HE (such as BFV Fan et al. (2012). BGV Brakerski et al. (2014). and CKKS Cheon et al. (2017). and bit-wise HE (such as FHEW Ducas et al. (2015). and TFHE Chillotti et al. (2020).) Among these schemes, Cheon-Kim-Kim-Song (CKKS) is regarded as the unique scheme to support homomorphic operations on float/complex number naturally. Therefore, CKKS could be utilized to construct a secure inference of phenotype from genotype.

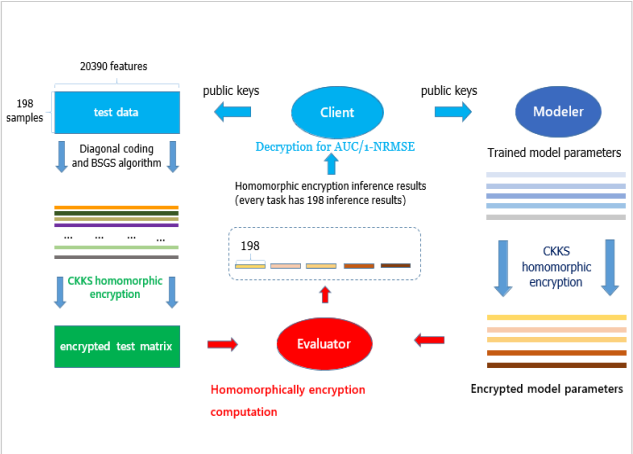
To make the secure inference efficiently, we propose an accurate and secure inference protocol on homomorphic encrypted genotype data with encrypted linear models. Firstly, scale the genotype data by feature importance with Xgboost or Adaboost then train a linear model to predict the phenotypes in plaintext. Secondly, encrypt the model parameters and genotype data with CKKS for secure inference. Thirdly, predict the phenotypes under CKKS homomorphic encryption. Finally, decrypt the encrypted predictions by client to compute the 1-NRMSE/AUC for model evaluation.

METHODS

Overview of the secure inference protocol

The secure inference consists of three parties: Client, Modeler, and Evaluator. 198 samples with 20390 features/variants are taken as an example to illustrate the details.

Figure 1: The flowchart of the secure inference protocol.



Step1: Client generates private key and public key and broadcasts the public keys to Modeler and Evaluator; **Step 2:** Client encrypts test data with the public key by diagonal coding and BSGS algorithm and CKKS homomorphic encryption then sends the encrypted results to Evaluator; **Step 3:** Modeler encrypts model parameters with received public key by CKKS homomorphic encryption then sends the encrypted results to Evaluator; **Step 4:** Evaluator executes homomorphically secure model inferences and sends the encrypted predictions to Client; **Step 5:** Client decrypts the ciphertext of predictions then the decrypted predictions are used for computing 1-NRMSE and AUC.

Firstly, Client generates private key and public key (for encryption), relinearization keys (for ciphertext multiplication) and Galois keys (for ciphertext rotation), and broadcasts these public keys to Modeler and Evaluator; Secondly, Client encrypts test data with the public key by diagonal coding, BSGS algorithm and CKKS homomorphic encryption and sends the encrypted results to Evaluator; Thirdly, Modeler encrypts model parameters with received public key and sends the encrypted results to Evaluator; Fourthly, after receiving the encrypted test data matrix and encrypted model parameters, Evaluator executes homomorphically secure model inferences with received relinerization keys and Galois keys, and sends the encrypted predictions to Client; Finally, Client decrypts the ciphertext of predictions then the decrypted predictions are used for computing $1-$ NRMSE and AUC.

Linear models with feature importance for predictions of phenotype from genotype in plaintext

$X \in \mathbb{R}^{m\times n}$ is the genotype matrix and X_{ij} denotes the j-th variant for i-th sample. $Y \in \mathbb{R}^{m\times K}$ is the phenotype matrix and Y_{ik} denotes the k-th phenotype for i-th sample.

Xgboost or Adaboost is used to obtain the feature importance for each phenotype then scale the raw genotype matrix by

$$X^{(k)} = X * diag(F_k)$$

Where F_k is the feature importance for the k-th phenotype.

If Y_k is the continuous phenotype, then the linear regression model should be

$$Y_k = X^{(k)} \times M_k + w_{0k} + \varepsilon_k$$

Where M_k is the linear regression parameter and $w0k$ is the intercept term. Let

$$W_k = M_k \times diag(F_k)$$

Then the final model is

$$Y_k = X \times W_k + w_{0k}$$

If Y_k is the category phenotype (i.e. 0 and 1), then the logistic regression model should be

$$P_k = \frac{1}{1+\exp\left(X^{(k)} \times M_k + W_{ok}\right)}$$

Where p_k is the probability of predicting the phenotype to be 1.

Similarity, the final model could be

$$p_k = \frac{1}{1+\exp(X \times W_k + w_{ok})}$$

In summary, both final models adopt the Linear Model with Feature Importance (LMFI) for phenotype inference.

CKKS Scheme

For a 2-power number N, we write $RN = \mathbb{Z}[X]/(X^N + 1)$ and $R_{N,q} = R_N/qR_N \equiv \mathbb{Z}q[X]/(X^N + 1)$. The lower-case letters with a "hat"symbol such as $\hat{a}$ represents some element in R_n, and a_j is denoted as the j-th coefficient of $\hat{a}$. The dot symbol $\cdot$ such as $\hat{a} \cdot \hat{b}$ is denoted as the multiplication of ring elements. We use bold lower- case letters symbol such as $\boldsymbol{a}$ to represent vectors, $\boldsymbol{a}[j]$ to represent the j-th component of $\boldsymbol{a}$, and $\boldsymbol{a} \| \boldsymbol{b}$ to represent the concatenation of vectors. Denote by $\boldsymbol{a} \ll k$ the left-hand-side rotation of the vector components. Denote by $\boldsymbol{a}^T\boldsymbol{b}$ the inner product of vectors and $\boldsymbol{a} \circ \boldsymbol{b}$ the Hadamard product of vectors, i.e., the element-wise multiplication. We use bold upper-case letters such as M to denote matrices, and $M[i, j]$ to denote the (i, j) entry of M.

As $\mathbb{Z}[X]/(X^N + 1)$ is isomorphic to $\mathbb{C}^{N/2}$, the ring structure allows us to encode a real vector $\boldsymbol{v} \in \mathbb{R}^l$ as a ring element of $R_{N,q}$ with $l \leq N/2$. The addition/multiplication in Rn corresponds to element-addition/multiplication of the real(complex) vector $\boldsymbol{v} \in \mathbb{R}^l$. Denote by $Encode(\boldsymbol{v}, \Delta) \in R_{N,q}$ and $Decode(\hat{v}, \Delta, l) \in \mathbb{R}^l$ the encoding of $\boldsymbol{v}$ with a scaling factor $\Delta > 0$, and the decoding of $\hat{v}$ with a scaling factor $\Delta > 0$ and a length $l > 0$ respectively.

The Ring Learning With Errors (RLWE) distribution $\text{RLWE}_s(N, q, \chi)$ with secret $s \in R_N$ and error distribution χ over R_N, produces pairs $(a, b) \in R_{N,q}$ where $a \leftarrow R_{N,q}$ is chosen uniformly at random, and $b = s \cdot a + e$ for $e \leftarrow \chi$.

The decisional Ring LWE assumption over R_N with error distribution χ, secret distribution χ' and m samples, states that when $s \leftarrow \chi'$, the product distribution $\text{RLWE}_s(N, q, \chi)^m$ is pseudorandom, i.e., it is computationally indistinguishable from the uniform distribution over $(R_{N,q} \times R_{N,q})^m$. As usual, χ' is the uniform distribution over $R_{N,3} = R_N/3R_N$ and χ is the discrete Gaussian distribution.

The security of CKKS scheme is based on RLWE Assumption. The following is the details of CKKS.

1.The key generation algorithm picks $s \leftarrow \chi', e \leftarrow \chi, a \leftarrow R_{N,q}$, and outputs secret key $sk = (-s, 1) \in R^2_{N,q}$ and public key $pk = (a, b) \in R^2_{N,q}$ where $b = s \cdot a + e$ follows the RLWE distribution.

2.The encryption algorithm, $Enc_{pk}(\hat{v})$ picks random $u \leftarrow \{0,1\}N$ and $e = (e_0, e_1) \leftarrow \chi 2$, and outputs $ct = u \cdot pk + e + (0, \hat{v}) \in R_{q,N}$, where $\hat{v} = Encode(\boldsymbol{v}, \Delta)$

3.The approximate decryption algorithm $Decsk(ct)$ outputs $\boldsymbol{v} = Decode(\hat{v}, \Delta)$ where $\hat{v} = \langle sk, ct \rangle \bmod q$.

By linearity of Enc, CKKS directly supports (bounded) addition of ciphertexts: if $ct_0 = (a_0, b_0)$ and $ct_1 = (a_1, b_1)$ are encryptions of $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ respectively, then the vector sum $ct_0 + ct_1 = (a_0 + a_1, b_0 + b_1) \bmod q$ is an encryption of $\boldsymbol{v}_0 + \boldsymbol{v}_1$. The plaintext-ciphertext multiplication is $\hat{v_0} \cdot ct_1 = (\hat{v_0} \cdot a_1, \hat{v_0} \cdot b_1) \bmod q$ is an encryption of $\boldsymbol{v}_0 \circ \boldsymbol{v}_1$, where $\hat{v_0} = Encode(\boldsymbol{v}, \Delta)$.

For homomorphic multiplication/rotation, extra public keys are needed. Denote by $EK/RotK$ the evaluation key for homomorphic multiplication/rotation, respectively.

Slot-wise Multiplication: Using EK, the product of two ciphertexts $ct_0 = (a_0, b_0)$, $ct_1 = (a_1, b_1)$ is computed as $Mul(ct_0, ct_1; EK) = ct_0 \times ct_1 = (a_0b_0 + a_1b_0, b_0b_1) + ReLin(a_0a_1; EK)$, where $ReLin(\alpha; EK) = (\alpha_0, \alpha_1)$ such that $\alpha_0 + \alpha_1 \cdot sk = \alpha \cdot sk^2 + e$ for some error e.

Rotation: Given the ciphertext ct which encrypts $Encode(\boldsymbol{v}, \Delta)$, an integer $k \in \mathbb{N}$, and a rotation key $RotK$, the operation $RotL^k(ct; RotK)$ results in an CKKS ciphertext that encrypts the left-hand-side rotated vector $Encode(\boldsymbol{v} \ll k, \Delta)$.

Rescale: Given the CKKS ciphertext ct which encrypts $Encode(\boldsymbol{v}, \Delta)$, and a factor $\Delta' \in \mathbb{R}$, the operation $Rescale(ct, \Delta')$ results in a ciphertext (with a smaller modulus) that encrypts $Encode(\boldsymbol{v}, \Delta/\Delta')$.

Self-repeating: $Decode(Encode(\boldsymbol{v} \| \cdots \| \boldsymbol{v}, \Delta), \Delta, l) = \boldsymbol{v}$. In other words, the encoding of some self-repeating vectors can be viewed as the encoding of a single copy.

Homomorphic Linear Evaluation

There are some existing approaches to homomorphic linear evaluation, i.e., the homomorphic multiplication of plain matrix and encrypted vector Halevi et al. (2014), Juvekar et al. (2018), Lu et al. (2021). The method proposed by Wenjie Lu et al. could cover both "tall" and "short" matrices efficiently and it is proper for our solution Lu et al. (2021). The following is the details of the matrix encoding and the process of homomorphic linear evaluation.

Algorithm 1. Matrix Encoding

Input: A plain matrix $M \in \mathbb{R}^{l \times n}$ with $l \leq n \leq N/2$ and a scaling factor Δ.

Output: elements $\{\hat{mi} \in R_N\}$ as encoded matrix.

1.Tiling and Diagonals. Define l vectors $\{m_j\}_{j=0}^{l-1}$ by going through the rows and columns of M

$$m_j[r] = M[r \bmod l, r + j \bmod n] \text{ for } r \in \{0,1,\cdots n\}$$

2.Let $g = \lceil\sqrt{l}\rceil$ and $b = \lceil l/g \rceil$, compute $\hat{m_i} = Encode(\boldsymbol{m}_{i_1 g+i_2} \gg i_1 g, \Delta)$ for

$i_1 \in \{0,1,\cdots, b-1\}$ and $i_2 \in \{0,1,\cdots, g-1\}$.

3.Output $\{\hat{\mathrm{m}_i}\}_{i=0}^{l-1}$

Algorithm 2. Homomorphic Linear Evaluation

Input: $ct_{\boldsymbol{z}} \in R_{N,q} \times R_{N,q}$ with $Encode(\boldsymbol{z}, \Delta_r)$ being encrypted. Rotation key $RotK$. Encoded matrix $\{\hat{m_i} \in R_N\}$ of M, which the column size is n. The scaling factor used to encode M is Δ.

output: ciphertext with encrypted $M \cdot \boldsymbol{z}$.

1.Let $g = \lceil\sqrt{l}\rceil$. For $i_2 \in \{0,1,\cdots, g-1\}$, compute $c_{i2} = RotL^{i2}(ct_{\boldsymbol{z}}; RotK)$.

2.Let b=[l /g]. Compute ct =

$$\sum_{i_1=0}^{b-1} RotL^{i_1 g}\left(\sum_{i_2=0}^{g-1} \hat{m}_{i_1 g+i_2} . c_{i_2}\right).$$

3.Let $\gamma = \log(n/l)$ and $ct_0 = ct$. Update iteratively for $1 \le j \le \gamma$

$$ct_j = RotL^{|2|j}(ct_j - 1) + ct_j - 1.$$

4.Output $Rescale(ct_\gamma, \Delta)$ as ct_{out}.

The rectangular matrix M is converted to a square matrix by repeating the rectangular matrix itself (called tiling) instead of expanding the rows (resp. cols) of M with zero-padding to be squared. A subset of the diagonals of the tiling matrix are constructed in Step 1 by looping through the rows and columns of M. This tiling is always possible without zero-padding because the number of rows and columns of M is always a power-of-2 value. The baby-step-giant-step (BSGS technique Cheon et al. (2019). in Step 2 of Algorithm 1 and Algorithm 2 aims to sum up some products of plaintext-ciphertext with a specific offset of homomorphic rotations. Specifically,

$$\sum_{j=0}^{l-1}\left(m_j \circ z \ll j\right) = \sum_{i_1=0}^{b-1}\left(\sum_{i_2=0}^{g-1}\left(\left(m_{i_1 g+i_2} \gg i_1 g\right) \circ \left(z \ll i_2\right)\right)\right) \ll i_1 g.$$

The term $m_{i_1 g+i_2} >> i_1 g$ is executed before Encode in CKKS since the cost of homomorphic rotations on encrypted $\boldsymbol{z}$ is expensive.

In Step 2 of Algorithm 2. ct can be viewed as a ciphertext, which corresponds to the sum of the l column vectors (each of size n/l), i.e., the matrix multiplication of each n/l columns of M and $\boldsymbol{z}$. The step 3 of Algorithm 2 aims to sum up the encrypted columns, resulting in ct_γ that encrypts a self-repeating vector $Encode\ (Mz \parallel \cdots \parallel Mz, \Delta r \Delta)$. It can just be viewed as $Encode\ (M\boldsymbol{z}, \Delta r \Delta)$ according to the property of the encoding function. Finally, the $Rescale(\cdot, \Delta)$ is used to reach the same scaling factor of $ct\boldsymbol{z}$.

The overall computation consists of l homomorphic multiplication and $g + b + \log n/l$ homomorphic rotations.

Modification and Optimization

For l and n not being power-of-2, we naturally extends the rows and columns of M with zero padding to fit the matrix encoding requirement. The expanded matrix has both the row size and the column size of being power-of-2.

For n larger than $N/2$, we split M into distinct submatrix $(M1, M2, \cdots, Ms)$, where $M1 \parallel M2 \parallel \cdots \parallel Ms = M$ and the column size of each submatrix is no more than $N/2$. Therefore, the encoding of M is converted into the encoding of s submatrix, which implies s homomorphic linear evaluations. According to the linearity of homomorphic rotation, homomorphic multiplications in each homomorphic linear evaluation are firstly executed and summed together followed by the execution of homomorphic rotations, The number of executing homomorphic rotations is reduced from s to only one.That is to say, the cost of homomorphic rotations are independent of the number of submatrices.

Assume only homomorphic linear evaluation is needed, the encoding of M can be improved by regarded two adjacent float rows as one complex row. That is, the same components of first row and second row are the real part and imaginary part of complex row respectively. With the improvement, the length of $\{\hat{m_j}\}$ is reduced by nearly half. After homomorphic linear evaluation, the ciphertext is the encryption of the complex vector which the real part and imaginary part of each component is the adjacent component of $M\boldsymbol{z}$. With the modification of matrix encoding, the time complexity and space complexity of homomorphic linear evaluation are almost reduced by half.

For homomorphic linear evaluation of encrypted matrix and encrypted vectors, the matrix is encrypted with $\{\hat{m_i}\}_{i=0^{l-1}}$, which are the outputs of Algorithm 1. Algorithm 2, $\sum_{i_2=0}^{g-1} \hat{m}_{i_1 g+i_2} . c_{i_2}$ is replaced by $\sum_{i_2=o}^{g-1} Mul\ (ct_{\hat{m}_{i_1 g+i_2}, c_{i_2}}, c_{i_2}; EK)$, where $ct_{\hat{m}_{i_1 g+i_2}}$ is the ciphertext of encrypted $\hat{m}_{i_1 g+i_2, c_{i_2}}$. Since the cost of $Relin\ (\cdot; EK)$ is expensive, we can sum up multiple ciphertext before $Relin$ in the case of $n > N/2$, i.e. multiple homomorphic linear evaluation of encrypted submatrix and encrypted sub-vector and ciphertext summation.

That is to say, if we need to compute multiple-multiplication-and-summation of between multiple $ct_{0,i} = (a_{0,i}, b_{0,i})$ and $ct_{1,i} = (a_{1,i}, b_{1,i})$, then $(a_0, {}_ib_0, {}_i + a_1,_ib_0,_i, b_0,_ib_1,_i)$ and $a_0,_ia_1,_i$ are firstly computed and summed up. As input, $\sum_i a_0, {}_ia_1, {}_i$ are executed by $Relin$ with EK secondly. Finally, the $\sum_i (a_0,_ib_0,_i + a_1,_ib_0,_i, b_0,_ib_1,_i)$ and $Relin(\sum_i a_0,_ia_1,_i ; EK)$ are summed up, which is equivalent to $\sum_i Mul(ct_{0,i}, ct_{1,i})$.

Note that $Relin$ is executed only once, and the cost of expensive $Relin$ is independent of the number of ciphertext pairs $((ct_{i,0}, ct_{i,1}))$.

Evaluation

The performance of the secure inference protocol is evaluated by the model accuracy and the running time. Specifically, the model accuracy is achieved by 1-NRMSE for continuous phenotype and AUC for category phenotype. Here NRMSE of the k-th phenotype is calculated by

$$\text{NRMSE}_k = \frac{\sqrt{\sum_{i=1}^{m}(y_{ik}-\hat{y}_{ik})^2/m}}{\max\{y_k\}-\min\{y_k\}}$$

Where $\hat{y}_{ik}$ is the prediction of the k-th phenotype for sample i. Let

$$S_k = \begin{cases} 1 - NRMSE_k & Y_k is continous \\ AUC_k & Otherwise \end{cases}$$

Then the average model accuracy could be inferred by

$$S = \frac{1}{k}\sum_{k=1}^{k} S_k$$

Both the average accuracy and running time are adopted to evaluate the performance of the secure inference protocol in the iDASH-2022 track2 challenge.

Datasets

We validate the secure inference protocol in a genotype dataset containing 3000 samples with 20390 variants. Each variant typically contains three genotypes such as AA, Aa and aa. Besides, all the samples contain 5 phenotypes, where three are continuous phenotypes and two are category phenotypes (i.e. 0 and 1). Furthermore, 198 samples are remaining for the blind test of the secure inference protocol. All these data are provided by the iDASH-2022 track2

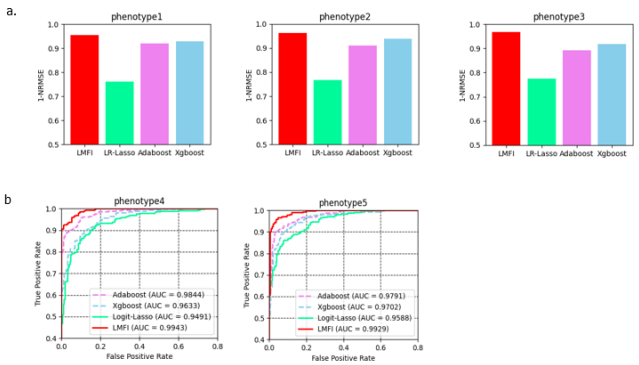
http://www.humangenomeprivacy.org/2022/.

RESULTS

LMFI outperform other methods in plaintext

To demonstrate the performance of LMFI in the inference of phenotype from genotype, we applied it to the dataset containing 5 phenotypes of 3000 samples with 20390 variants, and ten percent of the dataset is randomly selected as test data. LMFI achieves 0.9548, 0.9639, 0.9673, 0.9943, 0.9929 for overall phenotypes, respectively.

Figure 2: The comparisons of different methods. The performance of LMFI, Xgboost, Adboost, LR/Logit-lasso on continuous phenotypes (a) and category phenotypes (b).



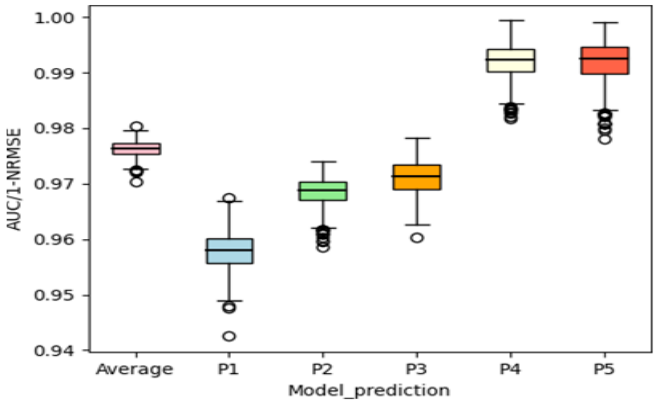
1-NRMSE(Normalized Root Mean Square Error) is used to evaluate the continous phenotypes and AUC is used to evaluate the category phenotypes. LMFI shows the best perfomance among these methods.

which performs much better than linear models with lasso in both continuous and category phenotypes. Furthermore, LMFI also shows better than non-linear models, such as Adaboost achieves 0.9185, 0.9151, 0.8936, 0.9844, 0.9791 and Xgboost achieves 0.9291, 0.9395, 0.9180, 0.9633, 0.9702.

The secure inference protocol is robust

To demonstrate the robust of the secure inference protocol, we applied it to a random sampling with 300 test samples for 100 times, and evaluated the performance by mean and standard deviation. The secure protocol could obtain 0.9577±0.0033 (mean and standard deviation), 0.9686±0.0025, 0.9711±0.0030, 0.9921±0.0029, 0.9920±0.0034 for 5 phenotypes, which indicating that the secure protocol is robust in different experiments.

Figure 3: The performance of the secure inference protocol.



a. The accuracy of the secure inference protocol in100 times of random sampling.

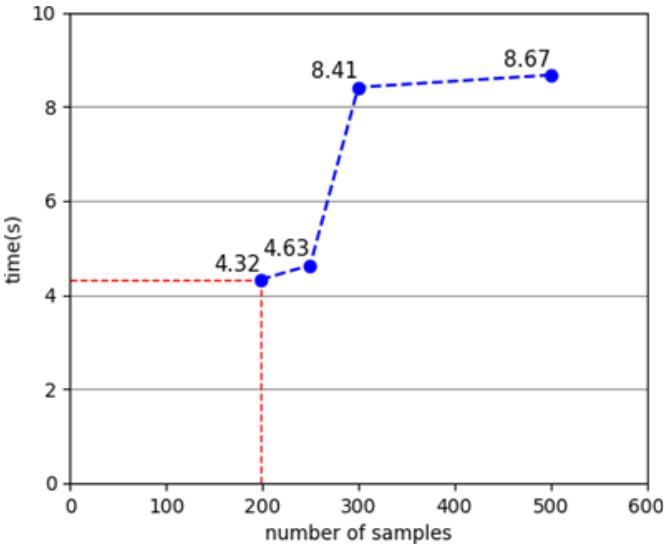
Furthermore, the average of the secure protocol achieves 0.9725 in a blind test with 198 samples.

The secure inference protocol is efficient

To demonstrate the efficient performance of the secure protocol, we test the performance with different sample size. Here we choose $N = 8192$, the coeff modulus is the multiplication of three primes with bit size 60, 60, 60 respectively by obeying the homomorphic encryption white paper Albrecht et al. (2021). For 198 × 20390 test data samples, which are regarded as 5 submatrix (with first four 198 × 4096 matrices and last 198 × 4006 matrix), we extend the five matrices to be 256 × 4096 matrices with zero padding. By assembling the adjacent rows of each submatrix into 128 × 4096 complex submatrix, we encode the complex submatrices to obtain 5 × 128 elements in R_N . In the process of homomorphic linear evaluation, 5 × 128 homomorphic multiplications and 16 + 8 + 5 homomorphic rotations are needed.

The protocol only takes 4.32s for 198 samples.

Figure 3: The performance of the secure inference protocol.



b. The running time of the secure inference protocol.

With the sample size arise, it does not change much (4.63s for 250 samples). Even with 500 samples, it takes 8.67s to obtain the final predications. All the computation are compiled by AMD EPYC 7K64@2.6GHz, running with 4 processes and the memory is 8GB.

DISCUSSION AND CONCLUSION

Construct an accurate, secure and efficient phenotype prediction model is essential for privacy and security computation. We have developed a secure inference protocol with encrypted linear models and it achieves good performance in the inference of phenotype and shows robust in 100 times of random sampling. Besides, it is very efficient and only takes seconds to predict the hundreds of samples. However, the protocol also needs to be further developed. Firstly, the protocol has not been test on more datasets. Secondly, the protocol should be further improved in homomorphic encrypted computation.

Thirdly, the linear models could be extended to non-linear models to achieve better accuracy and the homomorphic encrypted computation methods should be corresponded to be transformed. In conclusion, we have developed an accurate, secure and efficient phenotype prediction protocol and it takes only seconds to predict hundreds of samples.

ABBREVIATIONS

iDASH: integrating Data for Analysis, anonymization, and SHaring.

LMFI: Linear Model with Feature Importance.

NRMSE: Normalized Root Mean Square Error.

AUC: Area Under Curve.

HE: Homomorphic Encryption.

RLWE: Ring Learning with Errors.

BSGS: Baby-Step-Giant-Step.

CKKS: Cheon-Kim-Kim-Song.

DECLARATIONS

Supplementary information

Not appliable.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The datasets are available in the http://www.humangenomeprivacy.org/2022/ providing by the iDASH-2022 track2.

Competing interests

The authors declare that there is no conflict of interest.

Funding

The authors are supported by the Natural Science Foundation of China (NSFC) under Grants Nos. 11422108.

Acknowledgements

The authors acknowledge the grant NIH/NHGRI R13HG009072 that supported iDASH-2022 and co-organizer's efforts to prepare data/challenge.

Authors' contributions

G.Z. and M.Z. developed the inference model. F.Z. implemented the encryption of the model. M.Z., G.Z., F.Z and G.L. wrote the paper. All authors reviewed the manuscript.

REFERENCES

1.Benfer PN, Mitchell-Olds T. 2008. From genotype to phenotype: systems biology meets natural variation. Science. 320(5875):495-7.

2.Dowell RD, Ryan O, Jansen A, et al. 2010. Genotype to phenotype: a complex problem. Science. 328(5977):469.

3.Lehner B. 2013. Genotype to phenotype: lessons from model organisms for human genetics. Nature Reviews Genetics. Nat Rev Gene. 14(3):168-78.

4.Burton PR. 2007. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. Nature. 447(7145): p. 661-78.

5.Purcell S, Neale B, Todd-Brown K, et al. 2007. PLINK: a tool set for whole-genome association and population-based linkage analyses. Am J Hum Genet. 81(3):559-75.

6.Marchini J, Howie B, Myers S, et al. 2007. A new multipoint method for genome-wide association studies by imputation of genotypes. Nat Genet. 39(7):906-13.

7.Rakitsch B, Lippert C, Stegle O, et al. 2012. A Lasso multi-marker mixed model for association mapping with population structure correction. Bioinformatics. 29(2):206-14.

8.Waldmann P, Mészáros G, Gredler B, et al. 2013. Evaluation of the lasso and the elastic net in genome-wide association studies. Front Genet. 4:270.

9.Ogutu JO, Schulz-Streeck T, Piepho HP. 2012. Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. BMC Proc. 6(2): p. S10.

10.Banerjee R, Marathi B, Singh M. 2020. Efficient genomic selection using ensemble learning and ensemble feature reduction. Journal of Crop Science and Biotechnology. 23(4): p. 311-323.

11.Fan J, Vercauteren F. 2012. somewhat practical fully homomorphic encryption. Cryptology ePrint Archive.

12.Brakerski Z, Gentry C, Vaikuntanathan V. 2014. (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT). 6(3): p. 1-36.

13.Cheon JH. 2017. Homomorphic encryption for arithmetic of approximate numbers. in Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. Springer.

14.Ducas L, Micciancio D. 2015. FHEW: bootstrapping homomorphic encryption in less than a second. in Advances in Cryptology--EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34. Springer.

15.Chillotti I. 2020. TFHE: fast fully homomorphic encryption over the torus. Journal of Cryptology. 33(1): p. 34-91.

16.Halevi S, Shoup V. 2014. Algorithms in helib. in Advances in Cryptology– CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34. Springer.

17.Juvekar C, Vaikuntanathan V, Chandrakasan A. 2018. {GAZELLE}: A low latency framework for secure neural network inference in 27th {USENIX} Security Symposium ({USENIX} Security 18).

18.Lu WJ. 2021. PEGASUS: bridging polynomial and non-polynomial evaluations in homomorphic encryption in 2021 IEEE Symposium on Security and Privacy (SP). IEEE.

19.Cheon JH. 2019. Faster Linear Transformations in $\textsf {HElib} $, Revisited. IEEE Access. 7: p. 50595-50604.

20.Albrecht M. 2021. Homomorphic encryption standard. Protecting privacy through homomorphic encryption. p. 31-62.

