HEALER: Homomorphic computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS

Shuang Wang[1*§], Yuchen Zhang[1,2, §], Wenrui Dai[1,2], Kristin Lauter[3], Miran Kim[4], Yuzhe Tang[5], Hongkai Xiong[2], and Xiaoqian Jiang[1]

[1]Department of Biomedical Informatics, University of California, San Diego, CA, 92093.

[2]Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.

[3]Microsoft Research, San Diego, CA, 92122.

[4]Seoul National University, Seoul, 151-742, Republic of Korea.

[5]Department of EECS, Syracuse University, Syracuse, NY 13244.

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

SUPPLEMENTARY

Table S1. List of frequently used symbols in this paper

Symbols	Descriptions
$\boldsymbol{Y}$	A set of independent binary random variables
$\boldsymbol{y}$	$\boldsymbol{y} = (y_1, y_2, \dots, y_n)^T$ the realization of $\boldsymbol{Y}$ with n observations
n	Number of observations (records)
m	Number of categorical groups
p	A prime number serving as the modulus base
π_i	The response probability π_i for the i-th observation
x_i	The explanatory variable for the i-th observation
z_i	A nuisance variable may not be of direct interest
$\boldsymbol{d_i}$	A binary dummy vector for z_i
$\boldsymbol{\omega}, \boldsymbol{\beta}$	Model parameters with respect to the covariates $\boldsymbol{z}_i$ and $\boldsymbol{x}_i$
t_I	$t_I = \sum_{i=1}^{n} y_i x_i$ is the sufficient statistic of the explanatory variable
$t_{N,j}$	$t_{N,j} = \sum_{i=1}^{n} y_i d_{i,j}$ is the sufficient statistic by each category
r	Total number of samples
$\overline{\boldsymbol{Y}}$	Valid sample space
$\boldsymbol{\mathscr{y}}^*$	$\mathscr{y}^* = (\mathscr{y}_1^*, \mathscr{y}_2^*, \dots, \mathscr{y}_n^*)^{\mathrm{T}}$ random samples
$\widehat{\Delta}$	Encrypted version of variable or function Δ
$\hat{\mathscr{f}}(\hat{c})$	A homomorphic encrypted indicator function for $\mathrm{c} = 0$
M_{L_1}	A constant value in the $\hat{\mathscr{f}}(\hat{c})$ function
$\hat{\mathscr{h}}(\hat{c})$	A homomorphic encrypted indicator function for $\mathrm{c} > 0$
$\mathcal{M}_{\mathscr{k}}$	Precomputed constants in the $\hat{\mathscr{h}}(\hat{c})$ function
L_1, L_2	Upper bounds on ciphertext $\hat{c}$ in functions $\hat{\mathscr{f}}(\hat{c})$ and $\hat{\mathscr{h}}(\hat{c})$
L_s	Number of slots in ciphertext

S1. Sampling methods for solving exact logistic regression

The goal of exact conditional analysis is to evaluate how likely an observed response $\boldsymbol{y}$ is with respect to all possible responses $\{\boldsymbol{y}^*\}$ given sufficient statistics $\boldsymbol{t_N}$. It is computationally infeasible to scan all 2^n possible responses as the complexity increases exponentially with respect to the number of records n. For example, one would need to evaluate 2^{30} different $\boldsymbol{y}^*$ vectors in the case of $n = 30$. One workaround is to use sampling methods to approximate the permutation distribution of its sufficient statistics through a set of samples. In this paper, we only focus on building a secure exact logistic regression model with a single categorical nuisance covariate $z_i \in \{0, 1, \dots, m\}$ (m is the number of categories) using homomorphic encryption. When the nuisance variable $\boldsymbol{z}_i = (z_{i1}, z_{i2}, \dots z_{ih_1})$ includes multiple categorical covariates (i.e., $h_1 > 1$), the sampling distribution is more complicated. The problem can be tackled by using the Markov Chain Monte Carlo (MCMC) method (Mehta *et al.*, 2000) but it is out of the scope of this work.

To deal with a single categorical nuisance covariate $z_i \in \{1, \dots, m\}$, $m \geq 2$, we first introduce a binary dummy vector representation of z_i as $\boldsymbol{d_i} = (d_{i,1}, d_{i,2}, \dots, d_{i,m})$ with $d_{i,j} = \begin{cases} 1, & \text{if } j = z_i \\ 0, & \text{otherwise} \end{cases}$ (see Section S4 in this supplementary). Given the dummy coded representation, the sample space $\overline{\boldsymbol{Y}}$ can be represented as

$$\overline{\boldsymbol{Y}} = \{\mathscr{y}_1^*, \mathscr{y}_2^*, \dots, \mathscr{y}_n^*\colon \sum_{i=1}^{n} \mathscr{y}_i^* = t_0 \text{ and } \sum_{i=1}^{n} \mathscr{y}_i^* d_{i,j} = t_{N,j} \quad \text{for } j \in \{1,2,\dots,\mathrm{m}-1\}\} \qquad \text{(S1)}$$

where $t_{N,j} = \sum_{i=1}^{n} y_i d_{i,j}$ is the sufficient statistic contributed by each category $j, j \in \{1, \dots, m-1\}$. Here, we only need to consider $m-1$ degrees of freedom in Equation (S1) for m categories. The null distribution of $t_I = \sum_{i=1}^{n} y_i x_i$ can be estimated based on the samples uniformly drawn from $\overline{\boldsymbol{Y}}$. Denote $\boldsymbol{\mathscr{y}}^{(k)} = \left(\mathscr{y}_1^{(k)}, \mathscr{y}_2^{(k)}, \dots, \mathscr{y}_n^{(k)}\right)$ with $k = 1, \cdots, C_2$ as a total of C_2 samples uniformly drawn from $\overline{\boldsymbol{Y}}$ and denote $t_I^{(k)} = \sum_{i=1}^{n} \mathscr{y}_i^{(k)} x_i$ as its sufficient statistics for the parameter β. The p-value of exact test based on "conditional probability" (Mehta and Patel, 1995) is defined as the probability of getting the observed value of the test statistic, or a value with even greater evidence against the null hypothesis, can be calculated:

$$p_{val} = \frac{1}{C_2} \sum_{k=1}^{C_2} \mathbf{I}(t_I^{(k)} \geq t_I) = \frac{C_1}{C_2} \qquad \text{(S2)}$$

Here $\mathbf{I}\left(t_I^{(k)} \geq t_I\right) \in \{0,1\}$ is an indicator function for the comparison $t_I^{(k)} \geq t_I$ and $C_1 = \sum_{k=1}^{C_2} \mathbf{I}(t_I^{(k)} \geq t_I)$. The secure calculation of the p-value for β is the goal of our study. The estimation of the parameter β and the predictive inference of a response at x_i are also possible. But they are not the focus of this paper, we omit the details and interested readers can check the paper (Mehta and Patel, 1995).

S2. Secure rejection sampling

As we need to draw samples from the sample space defined in Equation (S1), a random sample $\boldsymbol{\mathscr{y}}^* = (\mathscr{y}_1^*, \mathscr{y}_2^*, \dots, \mathscr{y}_n^*)^T$ is considered as a valid permutation of $\boldsymbol{y} = (y_1, y_2, \dots, y_n)^T$ i.f.f. $\sum_{i=1}^n \mathscr{y}_i^* d_{i,j} = \sum_{i=1}^n y_i d_{i,j}$ for $\forall j \in \{1,2,\dots,m-1\}$. For example, in Supplementary Table S2, a valid permutation only takes place among y_i with the same color (i.e., in the same group). However, after homomorphic encryption, it is infeasible to differentiate different encrypted $\hat{\mathscr{y}}_i^*$ for $i = 1, 2, \dots, n$. For simplicity, we use $\hat{\Delta}$ to denote the encrypted version of variable or function Δ. To verify if a sample is valid over encrypted data, it is equivalent to compare whether all differences between $\hat{t}_{N,j} = \sum_{i=1}^n \hat{y}_i \hat{d}_{i,j}$ and $\hat{t}_{N,j}^* = \sum_{i=1}^n \hat{\mathscr{y}}_i^* \hat{d}_{i,j}$ for $\forall j \in \{1,2,\dots,m-1\}$ are zeroes. Let's denote $\hat{c}_j^* = \hat{t}_{N,j} - \hat{t}_{N,j}^*$ the difference in the j-th group. A negative integer c will be encoded by its p's-complement integer as $p - |c|$. For example, given $c \in [-L_1, L_1]$ and $p > 2L_1$, its p's-complement integer will be in the range $[0, L_1] \cup [p - L_1, p-1]$ for non-negative and negative plaintext c, respectively. Then, we can securely label (i.e., accept/reject) a sample based on the output of the following secure comparison function,

$$\hat{\mathcal{F}}(\hat{\boldsymbol{c}}^*) = \prod_{j=1}^{m-1} \hat{\mathcal{F}}_0(\hat{c}_j^*) = \begin{cases} \hat{1}, \text{if } c_j^* = 0, \forall j \in \{1,2,\dots,m-1\} \\ \hat{0}, \text{otherwise} \end{cases} \tag{S3}$$

where $\hat{\boldsymbol{c}}^* = (\hat{c}_1^*, \hat{c}_2^*, \dots, \hat{c}_{m-1}^*)^T$ is a vector of encrypted differences and the function $\hat{\mathcal{F}}_0(\hat{c}_j^*)$ can be defined as

$$\hat{\mathcal{F}}_0(\hat{c}_j^*) = \begin{cases} \hat{1}, \text{if } c_j^* = 0 \\ \hat{0}, \text{otherwise} \end{cases} \tag{S4}$$

Therefore, an encrypted output $\hat{1}$ can be used to label a valid sample. The Equation (S4) can be realized through the proposed secure integer comparison protocol as discussed in the supplementary Equations (S7) and (S8).

Algorithm 1: Homomorphic rejection sampling

```
0:  Inputs: encrypted vectors $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)^T$ and $\hat{\boldsymbol{d}}_j = (\hat{d}_{1,j}, \hat{d}_{2,j}, \dots, \hat{d}_{n,j})^T$ for $i = 1,2,\dots,n$ and $j = 1,2,\dots,m-1$
1:  Compute $\hat{t}_{N,j} = \sum_{i=1}^n \hat{y}_i \hat{d}_{i,j}$ for each group $j = 1,2,\dots,m-1$
2:  For each sample $k = 1,2,\dots,r$
3:    Draw $\hat{\boldsymbol{\mathscr{y}}}^{(k)} = \text{randperm}(\hat{\boldsymbol{y}})$ through random permutation of $\hat{\boldsymbol{y}}$
4:    For each categorical group $j = 1,2,\dots,m-1$
5:      Compute $\hat{t}_{N,j}^{(k)} = \sum_{i=1}^n \hat{y}_{s_i^*} \hat{d}_{i,j}$ with $* = k$
6:      Compute rejection criterion $\hat{c}_j^{(k)} = \hat{t}_{N,j} - \hat{t}_{N,j}^{(k)}$
7:      Evaluate $\hat{\mathscr{v}}_j^{(k)} = \hat{\mathscr{f}}(\hat{c}_j^{(k)})$ based on supplementary Equation (S8)
8:    end for
9:    Compute $\hat{\mathscr{v}}^{(k)} = \prod_{j=1}^{m-1} \hat{\mathscr{v}}_j^{(k)}$
10: end for
11: Outputs: all samples $(\hat{\boldsymbol{\mathscr{y}}}^{(1)}, \hat{\boldsymbol{\mathscr{y}}}^{(2)}, \dots, \hat{\boldsymbol{\mathscr{y}}}^{(r)})$ and the corresponding labels $(\hat{\mathscr{v}}^{(1)}, \hat{\mathscr{v}}^{(2)}, \dots, \hat{\mathscr{v}}^{(r)})$
```

Algorithm 1 (A1) summarizes the proposed homomorphic rejection sampling method. The inputs of the algorithm are encrypted vectors $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)^T$ and $\hat{\boldsymbol{d}}_j = (\hat{d}_{1,j}, \hat{d}_{2,j}, \dots, \hat{d}_{n,j})^T$ for $i = 1,2,\dots,n$ and $j = 1,\dots,m-1$. Before drawing samples, the algorithm first computes the observed sufficient statistic $\hat{t}_{N,j} = \sum_{i=1}^n \hat{y}_i \hat{d}_{i,j}$ contributed by each categorical group $j \in \{1,2,\dots,m-1\}$ in A1 line 1. Then, we draw a total of r independent samples through A1: lines 2 to 10. To generate the k-th sample $\hat{\boldsymbol{\mathscr{y}}}^{(k)}$, we define a random permutation function $\text{randperm}(\hat{\boldsymbol{y}})$ with encrypted $\hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)^T$ as input (see A1: line 3). The output of the function $\text{randperm}(\hat{\boldsymbol{y}})$ is a vector $\hat{\boldsymbol{\mathscr{y}}}^{(k)} = \hat{y}_{s_1^*}, \hat{y}_{s_2^*}, \dots, \hat{y}_{s_n^*}$, where $\boldsymbol{s}^* = (s_1^*, s_2^*, \dots, s_n^*)$ with $*= k$ is the k-th permutation instance of the original index vector $\mathcal{K} = (1,2,\dots,n)$. In A1 lines 4 and 8, we evaluate the label $\hat{\mathscr{v}}_j^{(k)}$ for each categorical group based on the difference $\hat{c}_j^{(k)}$ between observed sufficient statistic $\hat{t}_{N,j}$ and sample sufficient statistic $\hat{t}_{N,j}^{(k)}$. A1 line 7 securely evaluates the rejection criterion $\hat{c}_j^{(k)}$ based Equation (S8) as $\hat{\mathscr{v}}_j^{(k)} = \hat{\mathscr{f}}(\hat{c}_j^{(k)})$, where a valid or invalid sample $\hat{\boldsymbol{\mathscr{y}}}^{(k)}$ for a given categorical group j will be labeled as $\hat{1}$ or $\hat{0}$, respectively. In A1 line 9, we compute $\hat{\mathscr{v}}^{(k)} = \prod_{j=1}^{m-1} \hat{\mathscr{v}}_j^{(k)}$ as the label of a sample $\hat{\boldsymbol{\mathscr{y}}}^{(k)}$ over the sample space defined in Equation (S1), where we securely accept or reject a sample by labeling it as $\hat{1}$ or $\hat{0}$, respectively. Finally, the algorithm 1 outputs all samples $(\hat{\boldsymbol{\mathscr{y}}}^{(1)}, \hat{\boldsymbol{\mathscr{y}}}^{(2)}, \dots, \hat{\boldsymbol{\mathscr{y}}}^{(r)})$ and the related labels $(\hat{\mathscr{v}}^{(1)}, \hat{\mathscr{v}}^{(2)}, \dots, \hat{\mathscr{v}}^{(r)})$ in A1 line 11.

S3. Secure p-value computation

Given an encrypted sample $\hat{\boldsymbol{\mathscr{y}}}^{(k)} = (\hat{\mathscr{y}}_1^{(k)}, \hat{\mathscr{y}}_2^{(k)}, \dots, \hat{\mathscr{y}}_n^{(k)})^T$, one can evaluate the encrypted statistic as $\hat{t}_I^{(k)} = \sum_{i=1}^n \hat{\mathscr{y}}_i^{(k)} \hat{x}_i$. Based on Equation (S2), secure p-value calculation involves securely evaluating the indicator function $\boldsymbol{I}(t_I^{(k)} \geq t_I) \in \{0,1\}$ for the comparison $t_I^{(k)} \geq t_I$. Because it also requires the comparison over ciphertext, we need to build an equivalent secure indicator function as follows

$$\hat{\mathcal{H}}(\hat{c}) = \begin{cases} \hat{1}, & \text{if } c \geq 0 \\ \hat{0}, & \text{otherwise} \end{cases} \tag{S5}$$

where we denote $\hat{c} = \hat{t}_I^{(k)} - \hat{t}_I$ the encrypted difference between two encrypted sufficient statistics $\hat{t}_I^{(k)}$ and $\hat{t}_I$. We also use the p's-complement integer representation for a negative integer c as $p - |c|$. Given the definition of $c = t_I^{(k)} - t_I$, we can find that its p's-complement integer will be in the range $[0, L_2] \cup [p - L_2, p-1]$ with $L_2 \leq \frac{n}{2}$ and $p > 2L_2$. The equation (S5) can be rewritten as

$$\hat{\mathcal{H}}(\hat{c}) = \begin{cases} \hat{1}, \text{if } c \in [0, L_2] \\ \hat{0}, \text{if } c \in [p - L_2, p-1] \end{cases} \tag{S6}$$

Based on the similar idea used in the previous subsection, we can realize Equation (S6) through supplementary Equation (S11).

Algorithm 2: secure p-value computation

```
0:  Inputs: all samples $(\hat{\boldsymbol{\mathscr{y}}}^{(1)}, \hat{\boldsymbol{\mathscr{y}}}^{(2)}, \dots, \hat{\boldsymbol{\mathscr{y}}}^{(r)})$ and the corresponding labels $(\hat{\mathscr{v}}^{(1)}, \hat{\mathscr{v}}^{(2)}, \dots, \hat{\mathscr{v}}^{(r)})$ from the outputs of Algorithm 1; initialize two encrypted variables $\hat{c}_1$ and $\hat{c}_2$ as zeroes.
1:  Compute the observed sufficient statistic $\hat{t}_I = \sum_{i=1}^n \hat{y}_i \hat{x}_i$
2:  For each sample $\hat{\boldsymbol{\mathscr{y}}}^{(k)}$, $k = 1,2,\dots,r$
3:    Compute the sampled sufficient statistic $\hat{t}_I^{(k)} = \sum_{i=1}^n \hat{\mathscr{y}}_i^{(k)} \hat{x}_i$
4:    Compute the encrypted difference $\hat{c}^{(k)} = \hat{t}_I^{(k)} - \hat{t}_I$
5:    Evaluate $\hat{\mathcal{J}}^{(k)} = \hat{\mathscr{h}}(\hat{c}^{(k)})$ based on supplementary Equation (S11)
6:    Update the counters $\hat{c}_2 = \hat{c}_2 + \hat{\mathscr{v}}^{(k)}$ and $\hat{c}_1 = \hat{c}_1 + \hat{\mathscr{v}}^{(k)} \hat{\mathcal{J}}^{(k)}$ to
      accumulate the contribution from valid samples (i.e., $\mathscr{v}^{(k)} = 1$)
7:  end for
8:  Outputs: two counters $\hat{c}_1$ (i.e., encrypted numerator) and $\hat{c}_2$ (i.e., encrypted denominator) to user for $p$-value evaluation in Equation (S2)
```

Algorithm 2 (A2) summarizes the proposed secure p-value computation method. The inputs are all samples $(\hat{\boldsymbol{\mathscr{y}}}^{(1)}, \hat{\boldsymbol{\mathscr{y}}}^{(2)}, \dots, \hat{\boldsymbol{\mathscr{y}}}^{(r)})$ and the corresponding labels $(\hat{\mathscr{v}}^{(1)}, \hat{\mathscr{v}}^{(2)}, \dots, \hat{\mathscr{v}}^{(r)})$ from the outputs of Algorithm 1. In addition, we create two encrypted variables $\hat{c}_1$ and $\hat{c}_2$ with initial value as zeroes to count the number of ones in the indicator function $\boldsymbol{I}(t_I^{(k)} \geq t_I)$ and the number of valid samples, respectively. In A2 line 1, we compute the encrypted sufficient statistic $\hat{t}_I = \sum_{i=1}^n \hat{y}_i \hat{x}_i$ based on encrypted observations. For each sample $\hat{\boldsymbol{y}}^{(k)}$, $k = 1,2,\dots,r$, we compute the encrypted sufficient statistic

$\hat{t}_l^{(k)} = \sum_{i=1}^{n} \hat{y}_i^{(k)} \hat{x}_i$ based on the encrypted sample (A2 : line 3). Then, A2 line 4 securely computes the difference $\hat{c}^{(k)} = \hat{t}_l^{(k)} - \hat{t}_l$ between the sample and observed sufficient statistics. We evaluate the secure integer comparison function $\hat{J}^{(k)} = \hat{h}\left(\hat{c}^{(k)}\right)$ based on supplementary Equation (S11) in A2 line 5 and update the counters as $\hat{c}_2 = \hat{c}_2 + \hat{v}^{(k)}$ and $\hat{c}_1 = \hat{c}_1 + \hat{v}^{(k)}\hat{J}^{(k)}$ in A2 line 6. As only a valid sample has a non-zero label $v^{(k)} = 1$, both counters can exactly aggregate the contribution from valid samples. Finally, A2 line 8 outputs the two encrypted counters $\hat{c}_1$ (i.e., numerator) and $\hat{c}_2$ (i.e., denominator) to user for p-value evaluation in Equation (7).

The performance analysis of both algorithms in terms of acceptance rate, circuit depth and the number of homomorphic operations can be found in the discussion section in the main text and this supplementary.

S4. Examples of dummy vector representation

Table S2 shows an example of the dummy coding for a covariate with three categories. We use m dummy variables to represent a covariate with m categories in this setting (although an m-category covariate only have $m-1$ degrees of freedom) to facilitate the algorithm description in the main text.

Table S2. An example of the dummy coding scheme for the categorical covariate $z_i \in \{1,2,3\}$. A binary dummy vector with three elements $\boldsymbol{d_i} = (d_{i,1}, d_{i,2}, d_{i,3})$ is used to encode each group, where $d_{i,j}$ is non-zero if and only if $j = z_i$ and the same categorical group is shaded in the same color.

i	y_i	z_i	Dummy coding representation		
			$d_{i,1}$	$d_{i,2}$	$d_{i,3}$
1	0	2	0	1	0
2	1	3	0	0	1
3	1	1	1	0	0
4	0	1	1	0	0
5	1	3	0	0	1
6	0	3	0	0	1
7	1	2	0	1	0
8	1	2	0	1	0
9	0	1	1	0	0

S5. Homomorphic comparison of $\mathbf{c}_j^* = 0$

To realize Equations (S3) and (S4), we need to securely verify the equality $c_j^* = 0$ in function $\hat{\mathcal{F}}_0\left(\hat{c}_j^*\right)$ for $j = 1,2,\dots,m-1$. Previous studies (Cheon et al., 2015; Togan and Plesca, 2014; Ayday et al., 2013) demonstrated the feasibility of secure integer comparison over the finite field $GF(2^B)$. For example, the integer $a = 13$ over a finite field $GF(2^4)$ can be expressed as a binary vector $\boldsymbol{a_{bin}} = \left(a_{(1)}, a_{(2)}, a_{(3)}, a_{(4)}\right) = (1,1,0,1)$. Then, one can construct a secure function $\hat{g}\left(\hat{a}_{(l)}, \hat{b}_{(l)}\right) = \hat{a}_{(l)} + \hat{b}_{(l)} + \hat{1}$ for bit-wise comparison between the l-th encrypted bits $\hat{a}_{(l)}$ and $\hat{b}_{(l)}$ of two integers $\boldsymbol{a_{bin}}$ and $\boldsymbol{b_{bin}}$, where the output of $\hat{g}\left(\hat{a}_{(l)}, \hat{b}_{(l)}\right)$ is $\hat{1}$, i.f.f. $a_{(l)} = b_{(l)}$. However, a major problem of these methods is the high computational cost of the addition and multiplication operations over binary representation, as these operations require bit-wise operations, which need a deep circuit for large integers. In this section, we introduce an algorithm for secure integer comparison without using binary representation, by which addition and multiplication operations after comparison can be carried out directly over integers.

Based on the Wilson's Theorem (Silverman, 2006) $(p-1)! \equiv -1\ (mod\ p)$ with p as a prime number greater than 2, we can realize the secure comparison function $\hat{\mathcal{F}}_0\left(\hat{c}_j^*\right)$ in Equation (S4) with an encrypted integer input, where $c_j^* \in [0, L_1] \cup [p - L_1, p-1]$ as follow:

$$\hat{f}_0\left(\hat{c}_j^*\right) \equiv -\prod_{\ell=1}^{p-1}\left(\hat{\ell} - \hat{c}_j^*\right)\ (mod\ q) \tag{S7}$$

where q is the ciphertext modulus (i.e., a product of primes under double-CRT representation in the BGV scheme (Brakerski et al., 2012)). We can verify Equation (S7) in plaintext domain as $f_0\left(c_j^* = 0\right) = -(p-1)! \equiv 1\ (mod\ p)$. For $c_j^* \in [1, p-1]$ and $\ell = 1,2,\dots,p-1$ in Equation (S7), as one of the terms $\left(\hat{\ell} - \hat{c}_j^*\right)$ will yield a zero, the product in (S7) will be also zero with $c_j^* \in [1, p-1]$.

The evaluation of Equation (S7) could be very expensive, as it requires $p-1$ times homomorphic multiplications and p can be a large number. If the bounds of plaintext are known as $c_j^* \in [0, L_1] \cup [p - L_1, p-1]$, we can reformulate Equation (S7) in a more efficient way as

$$\begin{aligned}\hat{f}\left(\hat{c}_j^*\right) &\equiv -\left(\prod_{\ell=1}^{L_1}\left(\hat{\ell} - \hat{c}_j^*\right)\right)\left(\prod_{j=L_1+1}^{p-L_1-1}\hat{j}\right)\left(\prod_{i=p-L_1}^{p-1}\left(\hat{\imath} - \hat{c}_j^*\right)\right)(mod\ q)\\ &\equiv -\widehat{M}_{L_1}\left(\prod_{\ell=1}^{L_1}\left(\hat{\ell} - \hat{c}_j^*\right)\right)\left(\prod_{i=p-L_1}^{p-1}\left(\hat{\imath} - \hat{c}_j^*\right)\right)(mod\ q)\end{aligned} \tag{S8}$$

In Equation (S8), as $\widehat{M}_{L_1} = \prod_{j=L_1+1}^{p-L_1-1}\hat{j}$ is independent of ciphertext $\hat{c}_j^*$ given the bound L_1, we can precompute M_{L_1} as a constant value in our problem. For example, when considering the difference $c_j^* = t_{N,j} - t_{N,j}^*$ (see Section S2), we can find that L_1 is no larger than $\frac{n}{2}$.

S6. Homomorphic comparison of $\mathbf{c} \geq \mathbf{0}$

Based on the similar idea used in Equation (S8), we can realize Equation (S6) as follows with $p > 2L_2$.

$$\begin{aligned}&\hat{h}(\hat{c})\\ &\equiv \underbrace{-\left(\prod_{\ell=1}^{L_2}\left(\hat{\ell} - \hat{c}\right)\right)\left(\prod_{j=L_2+1}^{p-L_2-1}\hat{j}\right)\left(\prod_{i=p-L_2}^{p-1}\left(\hat{\imath} - \hat{c}\right)\right)}_{\text{Term } \hat{t}_0(\hat{c})}\\ &-\sum_{k=1}^{L_2}\underbrace{\left(\prod_{\substack{\ell=0\\ \ell\neq k}}^{L_2}\left(\hat{\ell} - \hat{c}\right)\right)\left(\prod_{j=L_2+1}^{p-L_2-1}\left(\hat{j} - \hat{k}\right)\right)\left(\prod_{i=p-L_2}^{p-1}\left(\hat{\imath} - \hat{c}\right)\right)}_{\text{Term } \hat{t}_k(\hat{c})}(mod\ q)\end{aligned} \tag{S9}$$

where q is the ciphertext modulus (i.e., a product of primes under double-CRT representation in the BGV scheme (Brakerski et al., 2012)). In Equation (S9), there are $L_2 + 1$ options to get an output of $\hat{1}$. It is easy to verify terms in plaintext as $t_0(c = 0) \equiv -(p-1)!\ (mod\ p) \equiv 1$ and $t_k(c = 0) \equiv 0$ due to the factor $\left(\prod_{\substack{\ell=0\\ \ell\neq k}}^{L_2}(\ell - c)\right)$. For $c \in [1, L_2]$, terms $t_0(c)$ and $t_{k\neq c}(c)$ always yield zeroes due to the factor $\prod_{\ell=1}^{L_2}(\ell - c)$ and $\prod_{\substack{\ell=0\\ \ell\neq k}}^{L_2}(\ell - c)$, respectively. The only non-zero term is $t_{k=c}(c) \equiv -c\prod_{\ell'=1}^{c-1}(\ell - c)\prod_{\ell''=c+1}^{p-1}(\ell'' - c) \equiv (p-1)!\ (mod\ p) \equiv -1$. Therefore, the output of $\hat{h}(\hat{c}) \equiv t_0(\hat{c}) - \sum_{k=1}^{L_2} t_k(\hat{c})$ is equal to $\hat{1}$ for $c \in [1, L_2]$. Based on the same idea, we can verify that the output of $\hat{h}(\hat{c})$ will be $\hat{0}$ for $c \in [p - L_2, p-1]$ due to the factor $\left(\prod_{i=p-L_2}^{p-1}(\hat{\imath} - \hat{c})\right)$. To reduce the number of homomorphic multiplication, we can simplify the Equation (S9) as

$$\hat{h}(\hat{c}) \equiv \left(\underbrace{\prod_{i=p-L_2}^{p-1} (\hat{\imath} - \hat{c})}_{\text{First term}}\right) \left(\underbrace{\left(\prod_{\ell=1}^{L_2} (\hat{\ell} - \hat{c})\right) + \sum_{k=1}^{L_2} \widehat{\mathcal{M}}_k \left(\prod_{\substack{\ell=0 \\ \ell \neq k}}^{L_2} (\hat{\ell} - \hat{c})\right)}_{\text{Second term}}\right) (mod\ q) \tag{S10}$$

where $\widehat{\mathcal{M}}_k = \left(\prod_{j=L_2+1}^{p-L_2-1} (\hat{j} - \hat{k})\right), k = 1,2, \dots, L_2$ are precomputed constants as they are independent of input $\hat{c}$ given L_2 and p. Noticing that the second term in Equation (S10) is a polynomial of degree L_2, we can further reduce its complexity by representing it with a recursive function as follows

$$\hat{h}(\hat{c}) \equiv \left(\underbrace{\prod_{\ell=p-L_2}^{p-1} (\hat{\ell} - \hat{c})}_{\text{First term}}\right) \mathcal{R}(\hat{c}, \lceil \log(L_2 + 1) \rceil, L_2 + 1, \widehat{\boldsymbol{\alpha}}) \tag{S11}$$

where $\mathcal{R}(\hat{c}, \lceil \log(L_2 + 1) \rceil, L_2 + 1, \widehat{\boldsymbol{\alpha}})$ is a recursive function defined as

$$\mathcal{R}(\hat{c}, \ell_1, \ell_2, \widehat{\boldsymbol{a}}) = \begin{cases} \hat{c}\hat{\alpha}_{\ell_2-1} + \hat{\alpha}_{\ell_2-2} & if\ \ell_1 \leq 1 \\ \mathcal{R}\left(\hat{c}, 0, \frac{\ell_2}{2^{\ell_1-1}}, \widehat{\boldsymbol{\alpha}}\right) + \sum_{\ell=2}^{\ell_1} \hat{c}^{2^{\ell-1}} \mathcal{R}\left(\hat{c}, \ell - 1, \frac{\ell_2}{2^{\ell_1-\ell}}, \widehat{\boldsymbol{\alpha}}\right) & otherwise \end{cases}$$

The ceiling function $\lceil \Delta \rceil$ returns the smallest integer not less than Δ. Here, $\widehat{\boldsymbol{a}} = (\hat{a}_{L_2}, \hat{a}_{L_2-1}, \dots, \hat{a}_0)$ is the coefficient vector of the polynomial after applying polynomial expansion on the second term in Equation (S10). For example, the recursive function $\mathcal{R}(\hat{c}, \log_2(L_2 + 1) - 1, L_2 + 1, \widehat{\boldsymbol{a}})$ for $L_2 = 7$ calculates the following

$$\underbrace{(\hat{c}^2)^2 \underbrace{\left(\hat{c}^2 \underbrace{(\hat{c}\hat{a}_7 + \hat{a}_6)}_{\mathcal{R}(\hat{c},\ell_1=1,\ell_2=8,\widehat{a})} + \underbrace{(\hat{c}\hat{a}_5 + \hat{a}_4)}_{\mathcal{R}(\hat{c},\ell_1=0,\ell_2=4,\widehat{a})}\right)}_{\mathcal{R}(\hat{c},\ell_1=2,\ell_2=8,\widehat{a})} + \hat{c}^2 \underbrace{(\hat{c}\hat{a}_3 + \hat{a}_2)}_{\mathcal{R}(\hat{c},\ell_1=1,\ell_2=4,\widehat{a})} + \underbrace{\hat{c}\hat{a}_1 + \hat{a}_0}_{\mathcal{R}(\hat{c},\ell_1=0,\ell_2=2,\widehat{a})}}_{\mathcal{R}(\hat{c},\ell_1=3,\ell_2=8,\widehat{a})}$$

where the coefficient vector $\widehat{\boldsymbol{a}}$ can be calculated based on the second term in Equation (S10) in advance. The benefit of representing $\hat{h}(\hat{c})$ with a recursive function is that we reuse partial results and balance the circuit depth and number of homomorphic multiplications. This will be discussed in more detail in Section S7.

Table S3. Complexity analysis in terms of cumulative circuit depth (CCD), # of homomorphic additions (HA) and # of homomorphic multiplications (HM) in Algorithm 1 (secure rejection sampling), where the # of HA and HM could be reduced by a factor of L_s using L_s-slot in SIMD parallel computation

Algorithm 1	CCD	# of HA	# of HM
1: $\hat{t}_{N,j} = \sum_{i=1}^{n} \hat{y}_i \hat{d}_{i,j}$ for $j = 1,2, \dots, m-1$	1	$(n-1)(m-1)$	$n(m-1)$
2: **For** $k = 1,2, \cdots, r$			
3: $\hat{\boldsymbol{y}}^{(k)} = \text{randperm}(\hat{\boldsymbol{y}})$	0	–	–
4: **For** $j = 1,2, \cdots, m-1$			
5: $\hat{t}_{N,j}^{(k)} = \sum_{i=1}^{n} \hat{y}_{s_i^k} \hat{d}_{i,j}$	1	$n-1$	n
6: $\hat{c}_j^{(k)} \leftarrow \hat{t}_{N,j} - \hat{t}_{N,j}^{(k)}$	1	1	–
7: $\hat{v}_j^{(k)} = \hat{f}\left(\hat{c}_j^{(k)}\right)$	$1 + \log(n)$	n	n
8: **end for**			
9: $\hat{v}^{(k)} = \prod_{j=1}^{m-1} \hat{v}_j^{(k)}$	$\log(2n(m-1))$	–	$m-2$
10: **end for**			
Total:	$\log(2n(m-1))$	$((2r+1)n - 1)(m-1)$	$((2r+1)n + r)(m-1) - r$

S7. Performance Analysis

As a supplementary to section 4.1, we provide the detailed performance analysis for Algorithms 1 and 2 in Tables S3 and S4, respectively. Table S3 analyzes the circuit depth and the number of homomorphic additions and multiplications in Algorithm 1 for secure rejection sampling. As shown in Table S3, a total number of $((2r + 1)n + r)(m - 1) - r$ homomorphic multiplications is required by Algorithm 1 to obtain r number of samples $\hat{\boldsymbol{y}}^{(k)}$ and their corresponding labels $\hat{v}^{(k)}$ with $k = 1, \dots, r$. Moreover, the circuit depth of the Algorithm 1 is also analyzed in Table S3. According to Equation (S8), given precomputed M_L, line 7 needs $1 + \log(n)$ levels based on optimized binary tree multiplication scheme as illustrated in Fig. S1. Accordingly, the levels for line 9 can also be derived as $\log(2n(m - 1))$. Since line 9 can be conducted independently for all samples, the circuit depth of Algorithm 1 is $\log(2n(m - 1))$.

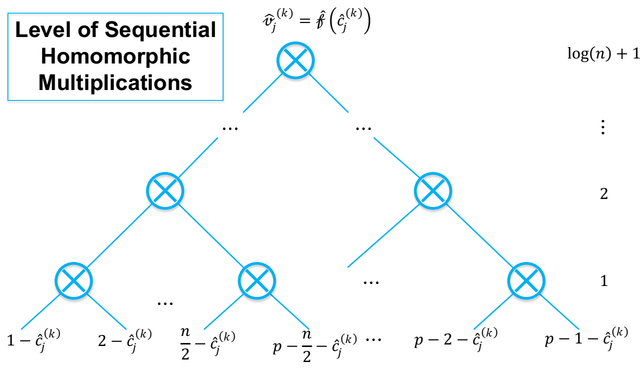

Level of Sequential Homomorphic Multiplications
$\hat{v}_j^{(k)} = \hat{f}\left(\hat{c}_j^{(k)}\right)$
$\log(n) + 1$
⋮
2
1
$1 - \hat{c}_j^{(k)}$
$2 - \hat{c}_j^{(k)}$
$\frac{n}{2} - \hat{c}_j^{(k)}$
$p - \frac{n}{2} - \hat{c}_j^{(k)}$
$p - 2 - \hat{c}_j^{(k)}$
$p - 1 - \hat{c}_j^{(k)}$

Fig. S1. Conceptual diagram for binary tree of product with $\left\{1 - \hat{c}_j^{(k)}, \cdots, \left(\frac{n}{2}\right) - \hat{c}_j^{(k)}, p - \left(\frac{n}{2}\right) - \hat{c}_j^{(k)}, \dots, p - 1 - \hat{c}_j^{(k)}\right\}$ for Algorithm 1 line 7.

Table S4. Complexity analysis in terms of cumulative circuit depth (CCD), # of homomorphic additions (HA) and # of homomorphic multiplications (HM) in Algorithm 2 (secure p-value computation)

Algorithm 2	CCD	# of HA	# of HM
1: $\hat{t}_I \leftarrow \sum_{i=1}^{n} \hat{y}_i \hat{x}_i$	1	$n-1$	n
2: **for** each $\hat{\boldsymbol{y}}^{(k)}, k = 1,2, \dots, r$			
3: $\hat{t}_I^{(k)} \leftarrow \sum_{i=1}^{n} \hat{y}_i^{(k)} \hat{x}_i$	1	$n-1$	n
4: $\hat{c}^{(k)} \leftarrow \hat{t}_I^{(k)} - \hat{t}_I$	1	1	–
5: $\hat{j}^{(k)} \leftarrow \hat{h}\left(\hat{c}^{(k)}\right)$	$1 + \log(n)$	n	$n + \log\frac{n}{2}$
6.1: $\hat{c}_2 \leftarrow \hat{c}_2 + \hat{v}^{(k)}$	$\log(2n(m-1))$	1	–
6.2: $\hat{c}_1 \leftarrow \hat{c}_1 + \hat{v}^{(k)}\hat{j}^{(k)}$	$\log(4n(m-1))$	1	1
7: **end for**			
Total	$\log(4n(m-1))$	$r(2n+2) + n - 1$	$r(2n + \log n) + n$

As shown in Table S4, Algorithm 2 compares the sufficient statistics $\hat{t}_I^{(k)}$ of each sample $\hat{\boldsymbol{y}}^{(k)}$ with $\hat{t}_I$, and aggregates valid count based on the label $\hat{v}^{(k)}$ of each sample. Line 5 in Algorithm 2 requires $1 + \log(n)$ levels for $n + \log\left(\frac{n}{2}\right)$ homomorphic multiplications by using the optimized multiplication scheme. As a result, given r samples $\hat{\boldsymbol{y}}^{(k)}, k = 1 \cdots, r$, the comparison of sufficient statistics proposed in Algorithm 2 requires a circuit depth of $\log(4n(m - 1))$, $r(2n + 2) + n - 1$ homomorphic additions and $r(2n + \log n) + n$ homomorphic multiplications.

It is worth mentioning that if the ciphertext base p is larger than 32, the above circuit depth analysis needs to be adjusted by a factor 2 for $32 < p < 2^{15}$.

S8. Computing environment and datasets descriptions

Our HEALER framework was implemented with the HElib, which is an open source homomorphic encryption library developed by the IBM research team (S. Halevi and Shoup). The performance was evaluated in three Ubuntu 12.04 virtual machines (VMs), each equipped with 96 GB memory and 8 threads, where two VMs have Intel(R) Xeon(R) CPU E7-4870 @ 2.40GHz and one VM has Intel(R) Xeon(R) CPU E7-4870 v2 @ 2.30GHz. The three VMs were hosted in the iDASH cloud (Ohno-Machado et al., 2012) at University of California, San Diego (UCSD). We tested the proposed HEALER protocol using three real rare Kawasaki Disease (KD) Coronary Artery Aneurysm (CAA) datasets with 15 or 30 records. These datasets are composed of 180, 372, and 744 SNPs, respectively, and one additional categorical nuisance variable (i.e., Percent C-reactive Protein (PCRP) expression level) with $m = 3$ categorical groups. Data in these KD datasets are from three different geographical locations including UCSD, University of Emory, and Genome Institute of Singapore. The PRCP values were grouped into three categories based on the following three regions $[-9, 9.0667)$, $[9.0667, 27.1333)$ and $[27.1333, 45.2]$. Fig. S2 shows the distributions of the case population and control population in these three PRCP categories. Moreover, the time and storage costs of key generation for above datasets are also described in Table S5.

Table S5. Parameters used in our experiments, where p is plaintext base; r' is lifting parameter for plaintext base; k' specifies the security level; L' is number of levels in modulus chain; c' is the number of columns in key switching matrix; and d' is hamming distance. Moreover, the time and storage costs of key generation are also provided as reference.

# of records	p	r'	k'	L'	c'	d'	L_s	Key generation	Public key size	Private key size
15	31	1	80	10	2	64	180	10.344s	69.3MB	70.4MB
30				11			372	10.697s	117 MB	118 MB

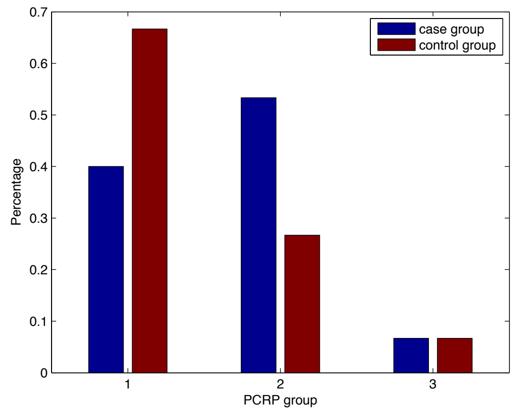


Fig. S2. Distributions of the case population (red bar) and control population (blue bar) in three different PRCP categories.

S9. Sensitivity of p-value when protecting research outcomes after computation using differential privacy

The p-value in exact logistic regression for the C_2 samples uniformly drawn from $\bar{\boldsymbol{Y}}$ is defined in Equation (S2), where the sample space $\bar{\boldsymbol{Y}}$ is defined in (S1). Let us define $D = \{(y_i, x_i, d_{ij}) : i \in \{1, \cdots, n\} \text{ and } j = \{1, \cdots, m\}\}$ as the set collecting all the $y_i \in \{0,1\}, , x_i \in \{0,1\}$ and $d_{ij} \in \{0,1\}$. Hence, we define the neighboring dataset of D with the same sample size.

Definition S1. $D' = \{(y_i', x_i', d_{ij}') : i \in \{1, \cdots, n\} \text{ and } j \in \{1, \cdots, m\}\}$ is called a neighboring dataset of D, when there exists $s \in \{1, \cdots, n\}$ such that $(y_s', x_s', d_{sj}') \neq (y_s, x_s, d_{sj})$ and $(y_i', x_i', d_{ij}') = (y_i, x_i, d_{ij})$ for $\forall i \neq s$.

To find the lower bound of sensitivity of p_{val}, we consider an extreme case for D and D'. Suppose $y = (1,0, \cdots, 0)$ and $x = (0,0, \cdots, 0)$ for D. The sufficient statistics t_I and $t_I^{(k)}$ are all zeroes, which means $p_{val}^{D} = 1$. From Definition S1, we set $y' = (1,0, \cdots, 0)$ and $x' = (1,0, \cdots, 0)$ in D', which derives that $t_I = 1$ and

$$t_I^{(k)} = \begin{cases} 1 & y^{(k)} = y' \\ 0 & otherwise \end{cases}. \tag{S12}$$

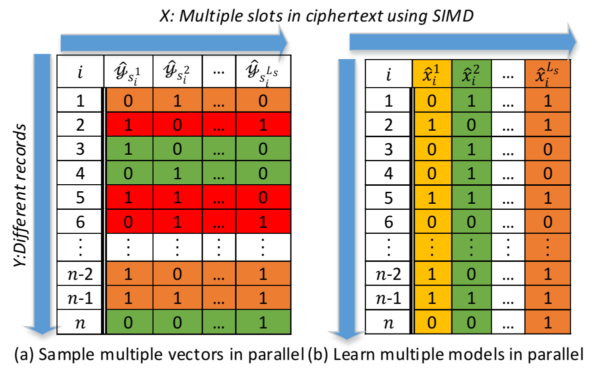
Thus, $p_{val}^{D'} = 1/|\bar{\boldsymbol{Y}}|$. According to equation (S1), given arbitrary $\boldsymbol{y}^* \in \bar{\boldsymbol{Y}}$, it satisfies $\sum_{i=1}^{n} y_i^* = t_0 = 1$. Therefore, the sensitivity is bounded by

$$\left|p_{val}^{D} - p_{val}^{D'}\right| = 1 - \frac{1}{|\bar{\boldsymbol{Y}}|} \leq 1 - \frac{1}{n}. \tag{S13}$$

In equation (S13), $|\bar{\boldsymbol{Y}}| = n$ can be achieved with proper d_{ij}'s. Thus, the lower bound of the sensitivity of p_{val} is $1 - 1/n$.

S10. Parallel Computation using multiple slots

The proposed HEALER framework also supports parallel computation using encryption schemes with ciphertext space $\mathbb{Z}_q^{L_s}$, which supports single instruction multiple data (SIMD) with L_s slots. It is worth mentioning that packing multiple ciphertexts into multiple slots have no impact on the size of encrypted data. Suppose $\hat{\boldsymbol{a}} = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{L_s})$ and $\hat{\boldsymbol{b}} = (\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_{L_s})$ are two encrypted ciphertext with L_s slots. Then, the addition $\hat{\boldsymbol{a}} + \hat{\boldsymbol{b}} = (\hat{a}_1 + \hat{b}_1, \hat{a}_2 + \hat{b}_2, \ldots, \hat{a}_{L_s} + \hat{b}_{L_s})$ and multiplication $\hat{\boldsymbol{a}} \cdot \hat{\boldsymbol{b}} = (\hat{a}_1 \cdot \hat{b}_1, \hat{a}_2 \cdot \hat{b}_2, \ldots, \hat{a}_{L_s} \cdot \hat{b}_{L_s})$ can be carried out in a SIMD manner in parallel. As shown in Fig. S3, we can utilize the multiple slots by packing: (i) pre-permuted vectors of the same observation $\hat{\boldsymbol{y}}$ or (ii) covariates from different models.



X: Multiple slots in ciphertext using SIMD
Y:Different records
(a) Sample multiple vectors in parallel
(b) Learn multiple models in parallel

i	$\hat{\mathcal{Y}}_{s_i^1}$	$\hat{\mathcal{Y}}_{s_i^2}$	…	$\hat{\mathcal{Y}}_{s_i^{L_s}}$
1	0	1	…	0
2	1	0	…	1
3	1	0	…	0
4	0	1	…	0
5	1	1	…	0
6	0	1	…	1
⋮	⋮	⋮	⋮	⋮
n-2	1	0	…	1
n-1	1	1	…	1
n	0	0	…	1

i	$\hat{x}_i^1$	$\hat{x}_i^2$	…	$\hat{x}_i^{L_s}$
1	0	1	…	1
2	1	0	…	1
3	0	1	…	0
4	0	1	…	0
5	1	1	…	1
6	0	0	…	0
⋮	⋮	⋮	⋮	⋮
n-2	1	0	…	1
n-1	1	1	…	1
n	0	0	…	1

Fig. S3. An example of packing data into multiple slots for SIMD computation, where (a) samples multiple vectors in parallel for the same model; (b) learns multiple models with different covariates (e.g., SNPs) in parallel

S11. Comparison with perturbation based protection methods

In this section, we provide additional results for the comparison between HEALER and perturbation based protection methods based on Differential Privacy (DP). To compare with the methods of applying DP before computation (DPBC) and DP after computation (DPAC), we select a KD dataset with 30 records and 744 SNPs. We selected three p-value cutoffs as 0.05, 0.01 and 0.005 to evaluate how many significant SNPs can be correctly preserved in the DPBC protected data (in terms of Recall and Precision) under different privacy budgets (i.e., $\varepsilon = 1$ and 0.5). The number of significate SNPs based on the raw data under different cutoffs are also provided in Table S6. Table S6 shows that the recalls of DPBC method are less than 0.05 for all setups, which means that less than 5% of the originally significant SNPs with p-value less than 0.05 can be preserved. For p-value cutoffs of 0.01 and 0.005, DPBC failed to preserve any originally significant SNPs. In contrast, the proposed HEALER framework can provide accurate results as well as protect the computation.

Table S6: Comparison between HEALER and DPBC methods in term of Recall and Precision in preserving significant SNPs with different p-value cutoffs 0.05, 0.01 and 0.005, and privacy budget $\varepsilon = 1$, and 0.5

p-value Cutoff	HEALER		Protection before computation				# of significant SNPs
			$\varepsilon = 1$		$\varepsilon = 0.5$		
	Recall	Precision	Recall	Precision	Recall	Precision	
0.05	1	1	0.0428	0.1034	0.0286	0.0556	70
0.01	1	1	0	0	0	0	11
0.005	1	1	0	0	0	0	5

Based on the idea of DPAC in (Yu and Ji, 2014), we also derived the corresponding DP algorithm for exact logistic regression in Section S9 in supplementary. Table S7 presents the comparison between HEALER and the DPAC methods. The results in terms of recall and precision in DPAC in Table 7 are better than these of DPBC in Table 6. However, when p-value cutoff is under 0.01, there is still no significant SNPs that can be preserved in DPAC. Tables 6 and 7 imply that it is hard to preserve significant SNPs after applying either DPBC or DPAC methods when record number is small.

Table S7: Comparison between HEALER and DPAC methods in term of Recall and Precision in preserving significant SNPs with different p-value cutoffs 0.05, 0.01 and 0.005, and privacy budget $\varepsilon = 1$, and 0.5

p-value Cutoff	HEALER		Protection after computation				# of significant SNPs
			$\varepsilon = 1$		$\varepsilon = 0.5$		
	Recall	Precision	Recall	Precision	Recall	Precision	
0.05	1	1	0.0714	0.2174	0.0571	0.1481	70
0.01	1	1	0	0	0	0	11
0.005	1	1	0	0	0	0	5

References

Ayday,E. *et al.* (2013) Privacy-Preserving Computation of Disease Risk by Using Genomic , Clinical , and Environmental Data. In, *Proceedings of USENIX Security Workshop on Health Information Technologies (HealthTech" 13)*.

Brakerski,Z. *et al.* (2012) (Leveled) fully homomorphic encryption without bootstrapping. In, *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12*. ACM Press, New York, NY, USA, pp. 309–325.

Cheon,J.H. *et al.* (2015) Homomorphic Computation of Edit Distance. In, *WAHC'15 - 3rd Workshop on Encrypted Computing and Applied Homomorphic Cryptography*.

Mehta,C.R. *et al.* (2000) Efficient Monte Carlo Methods for Conditional Logistic Regression. *J. Am. Stat. Assoc.*, **95**, 99–108.

Mehta,C.R. and Patel,N.R. (1995) Exact logistic regression: Theory and examples. *Stat. Med.*, **14**, 2143–2160.

Ohno-Machado,L. *et al.* (2012) iDASH. Integrating data for analysis, anonymization, and sharing. *J. Am. Med. Informatics Assoc.*, **19**, 196–201.

S. Halevi and Shoup,V. https://github.com/shaih/HElib.

Silverman,J.H. (2006) A friendly introduction to number theory. *Am Math Compet*, **10**, 12.

Togan,M. and Plesca,C. (2014) Comparison-based computations over fully homomorphic encrypted data. In, *Communications (COMM), 2014 10th International Conference on.*, pp. 1–6.

Yu,F. and Ji,Z. (2014) Scalable Privacy-Preserving Data Sharing Methodology for Genome-Wide Association Studies: An Application to iDASH Healthcare Privacy Protection Challenge. *BMC Med. Inform. Decis. Mak.*, **14**, S3.