Privacy-Preserving Statistical Analysis by Exact Logistic Regression

David A. duVerle*†, Shohei Kawasaki‡, Yoshiji Yamada§, Jun Sakuma‡¶ and Koji Tsuda*
* Graduate School of Frontier Sciences, University of Tokyo, 5-1-5 Kashiwa-no-ha, Kashiwa, Japan
† Email: dave@cb.k.u-tokyo.ac.jp
‡ Graduate School of SIE, University of Tsukuba, Tennodai 1-1-1, Tsukuba, Japan
§ Life Science Research Center, Mie University, Kurimamachiya-cho 1577, Tsu, Japan
¶ JST CREST

***Abstract*—Logistic regression is the method of choice in most genome-wide association studies (GWAS). Due to the heavy cost of performing iterative parameter updates when training such a model, existing methods have prohibitive communication and computational complexities that make them unpractical for real-life usage.**

We propose a new sampling-based secure protocol to compute exact statistics, that requires a constant number of communication rounds and a much lower number of computations. The publicly available implementation of our protocol (and its many optional optimisations adapted to different security scenarios) can, in a matter of hours, perform statistical testing of over 600 SNP variables across thousands of patients while accounting for potential confounding factors in the clinical data.

I. Introduction

In biomedicine, patient privacy and the safety of collected data is of utmost importance. Ideally, all such patient data can be collected and analysed in-house, however it is often necessary for an institution to outsource part of the work to larger commercial or public facilities with more advanced equipment. For example, new-generation sequencers can be used to call large numbers of single-nucleotide polymorphisms (SNPs) [1], identify diseases or prognosis, tailor treatments to patients [2] and ultimately develop new drugs [3]. Such sequencers can be unaffordable or simply not cost-efficient to maintain, and smaller institutions might prefer outsourcing to contractors, who then obtain knowledge of patients' genomic data.

While anonymised polymorphisms data (routinely made public in wide-scale genome studies [4]) are the subject of ongoing investigation about the degree of privacy they can guarantee [5], they are known to become extremely sensitive when tied to clinical information such as disease status, vital statistics or personal habits: such a tie greatly increases the chances of identification [5] and would hypothetically allow malicious parties to reveal a patient's health conditions from any easily-collected DNA sample. Even if the contracting institution is in good faith, data integrity remains a concern with any cloud-service providers and data breaches: a common occurrence [6], [7].

Our goal, in this paper, is therefore to produce useful statistical tests on patients' data vertically split (e.g. clinical and genomic) between two parties, without revealing either party's share to the other or tying them together in any way. We accomplish this by using homomorphic encryption to obliviously perform statistical testing on a type of non-asymptotic model. An approach rarely used in (non-oblivious) analysis of large-sized data, but with crucial advantages in an oblivious protocol.

We will first introduce our typical biomedical data analysis scenario in a non-oblivious context. Each instance (e.g., patient) is represented by three sets of variables: an outcome variable, explanatory variables and covariates. For instance, the outcome variable represents disease status, the explanatory variables represent SNP alleles and the covariates represent clinical information such as gender, age, ethnicity or smoking status. The outcome variable is assumed to be binary, separating the entire example set in case and control instances.

We would like to identify which explanatory variables are significantly relevant to the outcome. If the case and control sets have identical distribution of all clinical variables, simple correlation-based statistics such as chi-squared statistics can be used [8]. However, this is usually not the case: for example, if people of a particular ethnicity are overrepresented in either set, we may end up finding SNPs related to the ethnicity, not the condition. To avoid such shortcomings, covariates are usually controlled for by including them in logistic regression models [9].

Secure logistic regression models have already been proposed by [2], however, their protocol requires a very high number of communication rounds (proportional to the product of the number of SNPs by the number of examples, with a constant factor in the order of 10^3) and its computational complexity itself, while asymptotically close to linear (in the same variables) uses a very large constant factor, which makes the protocol's use unrealistic in real-life conditions, where non-negligible network latency and very large input size occur. This drawback is mainly due to complex computations involved in iteratively updating each parameter and costly approximations of real-number functions (e.g. using polynomials).

This paper presents a new approach employing *exact logistic regression* [10], [11], a model not implemented in GWAS packages such as PLINK [12], but commonly used in statistical and biological communities via tools such as SPSS [13] or R [14]. The term "exact" essentially means that the test

DOI 10.1109/SPW.2015.14

statistics are computed exactly without large-sample approximations, whereas the likelihood-based approach taken in [2] uses asymptotic statistics.

Crucially, the computation of exact statistics does not require parameter updates and can be done via sampling [15]. It only involves inner product calculations and a few comparisons, all of which are amenable to secure computation. In theoretical and experimental comparison to the best current alternative [2], we showed that our secure protocol based on exact logistic regression requires a (small) constant number of communication rounds and compares very favourably in empirical computational time.

II. Exact Logistic Regression

In this section, we first introduce exact logistic regression [10] in a traditional (non-oblivious) context. We wish to evaluate the association between binary outcome $y \in \{0,1\}$ and an explanatory variable $x_1 \in \{0,1\}$. In addition, a categorical covariate $x_2 \in \{0,\ldots,m\}$ is assumed to be known. The i^{th} example is a tuple $\{y_i, x_{1i}, x_{2i}\}$. Without loss of generality, the examples are assumed to be sorted with respect to x_2. Let us define q_k as the number of examples whose covariate is k ($x_{2i} = k$).

Using dummy coding for the categorical variable, the logistic model, with π denoting the probability of y being 1, is defined as

$$\log(\frac{\pi}{1-\pi}) = \gamma + \beta_1 x_1 + \sum_{k=1}^{m} \beta_{2k} [\![x_2 = k]\!],$$

where $[\![\mathcal{P}]\!] \in \{0,1\}$ is the boolean variable resulting from the evaluation of predicate $\mathcal{P}$.

Let vectors $\mathbf{y}, \mathbf{x_1}, \mathbf{x_{2k}}$ respectively denote the q-dimensional outcome, explanatory and k^{th} (out of m) covariate values of the q examples. Then, the sufficient statistics for γ, β_1 and β_2 are defined as $t_0 = \mathbf{1}^\top \mathbf{y}$, $t_1 = \mathbf{x_1}^\top \mathbf{y}$ and $t_{2k} = \mathbf{x_{2k}}^\top \mathbf{y}$, respectively.

In testing for statistical significance of the explanatory variable, we would like to find out if the observation $\mathbf{y}$ is special in that it gives particularly high correlation to $\mathbf{x_1}$. If the obtained level of explanatory correlation $\hat{t}_1$ is easily predictable from the existing information $\hat{t}_0$ and $\hat{t}_2$, it should not be regarded as statistically significant. In exact logistic regression, the sample space is defined as the set of all class vectors whose positive class size and covariate correlation are constrained to the observed value, called a *fiber* [16],

$$Y = \{\mathbf{y} \mid \mathbf{y}^\top \mathbf{1} = \hat{t}_0, \mathbf{y}^\top \mathbf{x_{2k}} = \hat{t}_{2k}, k = 1,\ldots,m\}.$$

It is equivalently represented as

$$Y = \{\mathbf{y} \mid \mathbf{y}^\top \mathbf{x_{2k}} = \hat{t}_{2k}, k = 0,\ldots,m\},$$

where $\hat{t}_{20} = \hat{t}_0 - \sum_{k=1}^{m} \hat{t}_{2k}$.

To calculate the p-value with respect to the explanatory variable, the null distribution of t_1 is derived by uniform sampling from Y. Sampling is done by concatenating $m+1$ vectors, each of which is a random permutation of a q_k dimensional vector composed of $\hat{t}_{2k}$ ones and $q_k - \hat{t}_{2k}$ zeros.

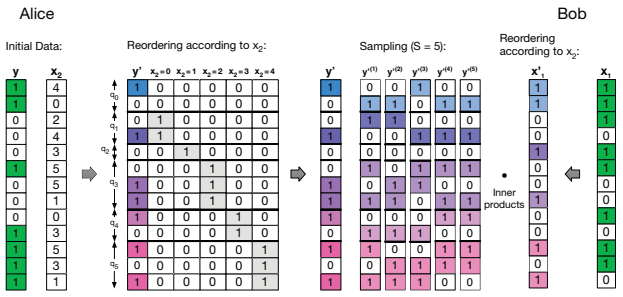

Fig. 1. Illustration of the Exact Logistic Regression protocol (without privacy): 1) Alice and Bob both reorder their respective input ($\mathbf{y} \in \mathbb{B}^{13}$) and ($\mathbf{x_1}$) according to the values of the covariate $\mathbf{x_2} \in \{0,1,2,3,4,5\}^{13}$ 2) Alice generates $S = 5$ samples with identical counts of positive values for each 'slice' of q_i rows corresponding to the same value i of $\mathbf{x_2}$ 3) They compute the inner products $t_1^{(i)}$ of $\mathbf{x'_1}$ with each $\mathbf{y'^{(i)}}$ and the inner product $\hat{t}_1$ of $\mathbf{x'_1}$ with $\mathbf{y'}$ 4) By counting the number of i (out of S) such that $t_1^{(i)} > \hat{t}_1$, they obtain the p-value. **Note:** in a privacy-preserving context, Bob cannot reorder $\mathbf{x_1}$ according to $\mathbf{x_2}$ (which Alice does not share with him) and all $\mathbf{y'^{(i)}}$ must therefore be re-ordered (by applying the inverse of the first ordering permutation) into actual $\mathbf{y^{(i)}}$ to match the original ordering of $\mathbf{y}$ and $\mathbf{x_1}$.

Let $\mathbf{y^{(i)}}, i = 1,\ldots,S$ denote the samples and $t_1^{(i)}$: each inner product $\mathbf{x_1}^\top \mathbf{y^{(i)}}$. Then, the p-value is computed as

$$\frac{1}{S} \sum_{i=1}^{S} [\![t_1^{(i)} \geq \hat{t}_1]\!]. \quad (1)$$

Figure II illustrates the method (using the reordered vectors directly, which does not affect the value of the inner products) for a single covariate $\mathbf{x_2}$ and $q = 13$ instances.

When there are multiple covariates, the fiber Y involves more constraints and sampling is more complex. Nevertheless, it can be achieved using Markov chain Monte Carlo methods [16], [15], [17]. It is also possible to use continuous variables [18].

For more details, refer to the textbook [11] and references therein.

III. Notations and Conventions

In all further protocols and discussions, *Alice* is the party with binary outcome ($\mathbf{y}$) and covariate ($\mathbf{x_2}$) data, while *Bob* has the explanatory variable data $\{\mathbf{x_{1i}}\}_i$. All cryptographic operations are conducted using Paillier's semi-homomorphically additive cryptosystem [19]. A single key pair is being used for all private- and public-key computations: this pair is generated by Alice and the public key sent to Bob once, at the beginning of the main protocol and assumed available in all sub-protocols. In the absence of possible confusion, all encryption operations, noted $E(x)$, and decryption operations, noted $D(c)$, implicitly use this key pair. We note $N = n^2$ the size of the resulting ciphertext domain in the Paillier cryptosystem ($\mathbb{Z}_N$) and all operations on ciphers are therefore implicitly $\mod N$ (omitted from pseudocode for brevity). We also adopt the common abuse of notation on plaintexts in $\mathbb{Z}_n$:

$$\forall p \in \mathbb{Z}_n, p \leq \frac{n-1}{2} : -p \equiv n - p \tag{2}$$

By convention, we write the i^{th} element of a vector $\mathbf{x}$ as $\mathbf{x}[i]$.

IV. Privacy-Preserving Exact Logistic Regression

Algorithm 1 outlines the entire protocol used by Alice and Bob to obliviously compute p-values out of input $\mathbf{y}$, $\mathbf{x_2}$ and $\{\mathbf{x_{1i}}\}_{i=1..s}$ without revealing either party's share of the data to the other party. By necessity, the common length of all three vectors, q, and the total number of explanatory variables, s, is known to both parties. Parameter S, also known to both, represents the number of samples used and can be adjusted according to the desired level of precision desired for the resulting p-value.

Algorithm 1 Privacy-Preserving Exact Logistic Regression

1: **procedure** PPELR
2: **Parameters:** number of samples used S
3: **Input from Alice:** $\mathbf{y} \in \mathbb{B}^q$ and $\mathbf{x_2} \in \mathbb{N}^q$
4: **Input from Bob:** $\mathbf{x_{1i}} \in \mathbb{N}^q$ for $i = 1, \ldots, s$
5: **Output to Alice:** p-values: $\mathbf{p_i}$ for $i = 1, \ldots, s$
6: ***Alice***:
7: **for** $i = 0, \ldots, q-1$ **do**
8: $\mathbf{y}[i] \leftarrow E(\mathbf{y}[i])$ ▷ Individually encrypt all elements of $\mathbf{y}$
9: **end for**
10: **for** $i = 1, \ldots, S$ **do**
11: $\mathbf{y^{(i)}} \leftarrow \pi_{\mathbf{x_2}}(\mathbf{y})$, where $\pi_{\mathbf{x_2}}$ is the random permutation outlined in II.
12: **for** $j = 0, \ldots, q-1$ **do**
13: $\mathbf{y}[j] \leftarrow$ REENCRYPT($\mathbf{y}[j]$) ▷ see IV-A1
14: **end for**
15: **end for**
16: ***Alice*** sends $(\mathbf{y}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(S)})$ to ***Bob***
17: ***Bob***:
18: **for** $i = 1, \ldots, s$ **do**
19: $\hat{t}_1 \leftarrow$ PRIVATESCALARPRODUCT($\mathbf{x_{1i}}, \mathbf{y}$)
20: **for** $j = \sigma(1), \ldots, \sigma(S)$ **do** ▷ σ is a random permutation in $[1, S]$
21: $t_1^{(j)} \leftarrow$ PRIVATESCALARPRODUCT($\mathbf{x_{1i}}, \mathbf{y^{(j)}}$)
22: $d_j \leftarrow t_1^{(j)} \cdot \hat{t}_1^{-1}$ ▷ homomorphic subtraction
23: GREATERTHANZERO(d_j) ▷ sub-routine returns α_{ij} to ***Alice***
24: **end for**
25: **end for**
26: ***Alice***:
27: **for** $i = 1, \ldots, s$ **do**
28: $p_i \leftarrow \frac{1}{S}\Sigma_j \alpha_{ij}$ ▷ each p-value is the sum of $\{\alpha_{ij}\}_j$ divided by the total number of samples S
29: **end for**
30: **end procedure**

The main steps of the protocol are:

- Lines 7 to 16: Alice prepares S sampled vectors (using the values of $\mathbf{x_2}$ to shuffle $\mathbf{y}$, in the way described in section II) and sends them to Bob, along with the original $\mathbf{y}$ vector. Each vector element of $\mathbf{y}$ is encrypted and the calls to REENCRYPT at line 13 guarantee that shuffled values are also encrypted and not identifiable across samples.
- Lines 18 to 21: For each possible explanatory variable $\mathbf{x_{1i}}$, Bob obliviously computes the inner product of $\mathbf{x_{1i}}$ with $\mathbf{y}$: $\hat{t}_{1i}$, and with each of the samples $\mathbf{y^{(j)}}$: $t_{1i}^{(j)}$, using the procedure described in algorithm 3. Samples are treated in random order, so as not to allow Alice to tie the result of the next step to a specific sample.
- Lines 22 to 23: For each explanatory variable and each sample, Bob uses the GREATERTHANZERO protocol described in algorithm 4 to replace the value of the encrypted difference $t_{1i}^{(j)} - \hat{t}_{1i}$ by a boolean indicator, α_{ij} (returned to Alice and deciphered by her), that only discloses whether the difference is greater than (or equal to) zero.
- Lines 27 to 29: Alice computes the p-values for each explanatory variable i, by counting the proportion of samples for which the sign indicator α_{ij} is 1.

A. Sub-Protocols

1) Re-Encryption: Given a plaintext p, one of the primitives most frequently used by our protocol is the generation of multiple encryptions of p in such a way that they are statistically indistinguishable in the IND-CPA model: that is, from a ciphertext $x = E(p)$, generate a new, indistinguishable, ciphertext y that decrypts to the same plaintext p. In a semantically secure cryptosystem such as Paillier, this is readily achieved by homomorphically adding a fresh encryption of zero, mathematically equivalent in Paillier to multiplying by an exponentiated random value, as shown in algorithm 2.

Algorithm 2 Re-Encryption Function

procedure REENCRYPT(x)
input: Encrypted value $x = E(p)$
output: Encrypted value $y = E(p)$ such that x and y are indistinguishable in the IND-CPA model
Choose a random $r \in \mathbb{Z}_N$
Return $x \cdot r^N \mod N$
end procedure

Proofs of correctness and security of this commonly used method can be found in e.g. [20].

2) Private Scalar Product: Algorithm 3 is a straightforward use of homomorphic addition to produce the Private Scalar Product [21] of two vectors (one of plain booleans and one of encrypted integers). Its correctness and security proofs are immediate consequences of Paillier's homomorphic properties and semantic security.

3) Oblivious Comparison to Zero: A central piece of our protocol relies on the ability to compare Ss pairs of encrypted values (equivalent to comparing each difference against zero).

Algorithm 3 Computing Private Scalar Product

procedure PRIVATESCALARPRODUCT($\mathbf{x}$, $\mathbf{y}$)
Input: $\mathbf{x} \in \mathbb{B}^q$
Input: $\mathbf{y} = [E(\mathbf{p}[i]), \ldots, E(\mathbf{p}[q])]$
Output: Ciphertext of the inner product: $t = E(\Sigma_i \mathbf{x}[i]\mathbf{p}[i])$
  $t \leftarrow 0$
  for $i = 0, \ldots, n-1$ **do**
    if $\mathbf{x}[i] = 1$ **then**
      $t \leftarrow t \cdot \mathbf{y}[j]$ ▷ homomorphically add $\mathbf{y}[j]$
    end if
  end for
  return t
end procedure

This should be done obliviously: without revealing the initial values or their difference to either party. By default, we also assume that only Alice should learn the result of the comparison test, so that no details about the resulting p-value are leaked (see section V-C for a relaxation of the protocol without this requirement).

Although many existing algorithms for oblivious comparison already exist in literature [22], [23], we introduced our own variant in algorithm 4, which takes advantage of two particular conditions of this protocol:

1) The values to be tested (p_c) are bound by a value known to both parties ($2q$) and many orders of magnitude smaller than the plaintext domain size ($q \ll n$).
2) Because the output of GREATERTHANZERO is used in a statistical context, to evaluate p-values through sampling, it is perfectly acceptable for the algorithm to be non-deterministic and output its result with a (small) error rate.

Algorithm 4 offers an efficient method that takes advantage of these two points to provide a sign test in a single round and a number of operations proportional to the bit-size of the input, with a bounded, arbitrarily small, chance of error (see theorem 1 and its proof).

The algorithm can be broken up into three parts:

Lines 5 to 7 reduce the original problem (oblivious comparison of two encrypted values) to an instance of Yao's "millionaires problem" [24], which has many known efficient solutions [25], [26]. The last part of our algorithm implements one such solution, based on the GT-SCOT protocol proposed in [22]. However, because current algorithms for Yao's millionaires problem tend to have a complexity in the domain of the values to compare, we can greatly reduce the computational cost by noting that the two numbers only differ in their last d' bits (where $d' \in \mathcal{O}(\log p_c)$). Lines 9 to 15 therefore compute d' before initiating the GT-SCOT protocol on the d' least significant bits from each value.

To prove the overall correctness of algorithm 4, we first prove the correctness of our reduction and restriction to d' bits:

Theorem 1. *Using the same notations and conditions as*

Algorithm 4 Obliviously revealing whether a value is greater than zero

1: **procedure** GREATERTHANZERO(c)
2: **Input from Bob:** $c = E(p_c)$
3: **Output to Alice:** $\alpha \in \mathbb{B}$, such that $\alpha = 1 \iff p_c \geq 0$ with arbitrarily low error rate.
4: ***Bob***:
5: $c \leftarrow c \cdot c \cdot E(1)$ ▷ $p_c \leftarrow E(2p_c + 1)$
6: $x \leftarrow c \cdot E(r)$ with $r \in \mathbb{Z}_n$ random.
7: ***Bob*** sends x to ***Alice***.
8: ***Alice***:
9: $p_x \leftarrow D(x)$
10: $d \leftarrow \lceil (\log_2 q) \rceil + 1$ ▷ upper bound on bit-length of p_c
11: $d' \leftarrow d + 1$
12: **while** $p_x \mod 2^{d'} = 0$ **do**
13: $d' \leftarrow d' + 1$
14: **end while**
15: $d' \leftarrow d' + 1$
16: ***Alice*** sends $[b_0, \ldots, b_{d'-1}] = [E(p_x \mod 2^{d'}), E(p_x \mod 2^{d'-1}), \ldots, E(p \mod 2)]$ to ***Bob***
17: ***Bob***:
18: $[a_0, \ldots, a_{d'-1}] = [r \mod 2^{d'}, r \mod 2^{d'-1}, \ldots, r \mod 2]$
19: **for** $i = 0, \ldots, d'-1$ **do**
20: **if** $a_i = 0$ **then**
21: $d_i \leftarrow b_i$ ▷ $d_i \leftarrow E(D(b_i) - a_i)$
22: $f_i \leftarrow b_i$ ▷ $f_i \leftarrow E(a_i \oplus D(b_i))$
23: **else**
24: $d_i \leftarrow b_i \cdot E(-1)$ ▷ $d_i \leftarrow E(D(b_i) - a_i)$
25: $f_i \leftarrow E(1) \cdot d_i^{-1}$ ▷ $f_i \leftarrow E(a_i \oplus D(b_i))$
26: **end if**
27: **if** $i = 0$ **then**
28: $g_i \leftarrow 0$
29: **else**
30: $g_i \leftarrow g_{i-1} \cdot g_{i-1} \cdot f_i$ ▷ $g_i \leftarrow E(D(2D(g_{i-1}) + D(f_i)))$
31: **end if**
32: $t_i \leftarrow d_i \cdot (g_i \cdot E(-1))^{r_i}$, with a randomly chosen $r_i \in \mathbb{Z}_n$. ▷ $t_i \leftarrow E(D(d_i) + D(r_i)(D(g_i) - 1))$
33: **end for**
34: ***Bob*** randomly shuffles all t_i and sends $\{t_{\sigma(0)}, \ldots, t_{\sigma(d'-1)}\}$ to ***Alice***.
35: ***Alice***:
36: $\alpha \leftarrow 0$
37: **for** $i = 0, \ldots, d'-1$ **do**
38: **if** $D(t_i) = 1$ **then**
39: $\alpha \leftarrow 1$ ▷ $\alpha = 1 \iff \exists i, D(t_i) = 1$
40: **end if**
41: **end for**
42: **end procedure**

algorithm 4: $[\![p_c \geq 0]\!] = [\![r \geq p_x]\!]$*, with probability at least* $(1-\epsilon)$.

Proof. First we note that line 5 does not affect $[\![p_c \geq 0]\!]$ (p_c becomes strictly positive if $p_c = 0$ and remains of the same sign otherwise).

We consider the relation between p_c, r and $p_x = p_c + r$ at line 6:

- If $p_c \leq \frac{n-1}{2}$ ($\equiv p_c \geq 0$), then $p_x \geq p_c$, except if $r \in [n - p_c, n]$ (causing a modulo wrap-around) which happens with probability: $\epsilon = \frac{p_c}{n}$.
- Conversely, if $p_c > \frac{n-1}{2}$ ($\equiv p_c < 0$), then $p_x < p_c$ (due to the modulo wrap-around), also with error probability: ϵ.

□

Furthermore, because $\epsilon < \frac{1}{2}$ (in practice $\epsilon \ll \frac{1}{2}$), we can easily see that a trivial modification of algorithm 4 using k pairs (r_i, p_{c_i}) would allow arbitrarily small overall error rate.

For practically any applications of our protocol, however, we have $p_c \ll n$ (e.g. $\log_2(n) \geq 1024$ and $\log_2(p_c) \leq 10$) and the error rate becomes negligible compared to the sampling error of our exact logistic regression test.

Lemma 1. *Using the same notations and conditions as algorithm 4:* $[\![r \geq p_x]\!] = [\![LSB_{d'}(r) \geq LSB_{d'}(p_x)]\!]$*, where* $LSB_i(x)$ *represents the* i *least significant bits of* x.

Proof. This lemma is a direct application of bitwise arithmetic: we know that the difference between r and p_x is of length at most $(\log_2(p_c) + 1) \leq d$ bits. In order to account for the possibility of carry-on (ones in r resulting in zeros in p_x, after the $d+1$ least significant bits), we increment d' until we pass at least one non-zero bit. Every bit after that will be guaranteed to be identical between the two values and would not affect the comparison. □

Detailed proofs of correctness and security of algorithm 4's resolution of Yao's millionaires problem in lines 16 to 41 can be found in the original publication [24], so we will only briefly outline the method here:

For each bit at the same position, i, in the two values to compare: $d_i \in \{-1, 0, 1\}$ will contain the difference between the two bits and $f_i \in \mathbb{B}$: the result of a bitwise *exclusive or* operation between the two bits. Since the bits are ordered from most to least significant, the sign of their difference will be decided by the first position k where the bits differ ($f_k = 1$) and will be the same as d_k. We can observe that the sequence of g_i will contain a number of zeros, followed by a single 1 at position k, followed by values > 1 afterward. Therefore, t_i will contain a randomly obfuscated value for all i except $i = k$, where $g_i - 1 = 0$.

Using the above theorems and lemma, we can directly deduce theorem 2.

Theorem 2. *Using the notations and input outlined in algorithm 4, we have output:* $\alpha = [\![p_c \geq 0]\!]$ *and algorithm 4 is correct.*

Additionally, we can prove theorem 3:

Theorem 3. *Assuming semantic security of Paillier's cryptosystem, in the semi-honest adversary model, algorithm 4 does not reveal any new information to either party, except for Alice's output of* $\alpha \equiv [\![p_c \geq 0]\!]$.

Proof (sketch). We can easily see that none of the ciphertexts sent by Bob to Alice reveal any information about the input data (obfuscation by random values on lines 7 and 32), except for one single value $\alpha = t_k \in \{-1, 0, 1\}$. Additionally, the case $t_k = 0$, which would reveal to Alice that p_c is equal to 0, is removed by line 5. Hence the only information revealed to Alice by Bob's encrypted communications is $\alpha \equiv [\![t_k = 1]\!]$.

All elements in the bit-decomposition sent by Alice to Bob on line 16 are encrypted with Alice's key and do not leak any information, except for the length of the array: d'. However, d' depends on the value of bits $d+1$ to d' of x (where $d \equiv \lceil(\log_2 q)\rceil + 1$ is a parameter of the input, known to both parties), which can be deduced by Bob from looking at the same positions in r (and therefore d' is already known to Bob). Therefore, Bob learns no new information from the protocol. □

Although this is not mandated by the particular needs of our statistical test, we note that it would be possible to modify algorithm 4 to make it entirely deterministic, using Nishide *et al.*'s suggested 'LSB Protocol for Special Case of Interval Test Protocol' [23]. However, the resulting protocol would require a much larger (constant) number of communication rounds and homomorphic operations than at present.

B. Proof of Correctness

Theorem 4. *Values* $\{\alpha_i\}_i$ *returned by algorithm 1 are p-values for the exact logistic regression test outlined in section II, using input* $\mathbf{y}$*,* $\mathbf{x_{1i}}$ *and* $\mathbf{x_2}$.

Proof. Given the homomorphic properties of the Paillier cryptosystem (and the correctness of the REENCRYPT procedure in section IV-A1), it immediately follows that the vectors sent on line 16 correctly encrypt each element of the plaintext vectors $\mathbf{y}$ and $\mathbf{y^{(i)}}$ described in section II.

Similarly, the correctness of PRIVATESCALARPRODUCT guarantees that $\hat{t}_1$ and $t_1^{(j)}$ on lines 19 and 21 are correct encryptions of the same-named variables in section II.

Based on theorem 2, we show that the returned values α_i match the values $[\![t_1^{(i)} \geq \hat{t}_1]\!]$ that appear in formula 1 and it immediately follows that line 39 produces the p-value, such as computed in formula 1. □

C. Proof of Security

Theorem 5. *Assuming semantic security of Paillier's cryptosystem, in the semi-honest adversary model, through running algorithm 1, Bob does not learn anything about Alice's data (*$\mathbf{y}$ *and* $\mathbf{x_2}$*) and Alice learns nothing on* $\mathbf{x_1}$ *other than the total number of samples for which the difference of inner-products (*$\mathbf{x_1}^\top \mathbf{y^{(i)}} \geq \mathbf{x_1}^\top \mathbf{y}$*) is positive.*

Proof (sketch). Proof of security for algorithm 1 is a direct consequence of the proven security of the different sub-protocols used (see sections IV-A1 to IV-A3):

Outside of protocol GREATERTHANZERO, the only data exchanged between the two parties is the list of sampled vectors sent by Alice on line 16. Because all elements are individually encrypted/re-encrypted, they do not leak any information about the input data.

Based on theorem 3, we know that the α_i returned to Bob at the end of the protocol do not leak any information beside the sign of the inner product difference. The random permutation on line 20 ensures that individual α_i cannot be tied to a specific sample and therefore Alice only learns the total number of samples for which the difference is positive. □

D. Complexity Analysis

1) Communication Rounds: Although we adopted a more granular notation for clarity, all calls to GREATERTHANZERO at line 23 and nested exchange inside the procedure, are independent of one another and can be merged into a single parallel call totalling one communication round (inside GREATERTHANZERO), keeping the total number of communication rounds for the entire protocol constant and equal to 2.

2) Communication Complexity: The total numbers of ciphertexts (of length $\log_2(N)$ bits) exchanged during these two rounds are:

- $(S+1)q$ on line 16 from Alice to Bob.
- Ss on line 7 of GREATERTHANZERO from Bob to Alice.
- Ssd' on line 16 of GREATERTHANZERO from Alice to Bob. Because d' depends on the occurrence of a 1 bit (after the d^{th} bit) in a random sequence, we can easily see that (on average) $d' = \log_2 q + 5$: we have at most $Ss(\log_2 q + 5)$ ciphertexts in total.
- Finally, Bob sends back: $Ss(\log_2 q + 5)$ ciphertexts to Alice.

The total size of the messages exchanged is therefore: $(13Ss + (S+1)q + 2Ss\log_2 q)\log_2 N \in \mathcal{O}((s + q + s\log q)\log N)$.

3) Computational Complexity: To estimate the computational cost of the protocol, we enumerate the number of exponentiations, $\mathcal{E}$, and multiplications, $\mathcal{M}$, in $\mathbb{Z}_N$ (the few operations on plaintext integers have comparatively negligible cost):

- q invocations of Paillier's encryption procedure for the initial set-up on Alice's side: $q(2\mathcal{M} + \mathcal{E})$.
- Sq calls to REENCRYPT to hide the values of the shuffled vectors: $Sq(\mathcal{M} + \mathcal{E})$.
- For each of $\mathbf{x_{11}}, \ldots, \mathbf{x_{1s}}$, we have:
  - $S+1$ calls to PRIVATESCALARPRODUCT$(\mathbf{x}, \mathbf{y})$ ($|\mathbf{x}| = |\mathbf{y}| = q$): $(S+1)q\mathcal{M}$.
  - S multiplications: $S\mathcal{M}$.
  - S executions of GREATERTHANZERO (with $d' = q + 5$): $S\log_2 q(10\mathcal{M} + 4\mathcal{E}) + 55\mathcal{M} + 21\mathcal{E}$

fast-reencrypt: - (log q + 5) (M+E) - Sq E

The total number of operations in $\mathbb{Z}_N$ is therefore:

$$\begin{aligned} T = &[(S+1)\cdot sq + 10S\cdot(s\log_2 q) + 56S\cdot s]\mathcal{M} \\ &+[4S\cdot s\log_2 q + (1+S)\cdot q + 21S\cdot s]\mathcal{E} \end{aligned} \quad (3)$$

and $T \in \mathcal{O}(sq\mathcal{M} + (s\log q + q)\mathcal{E})$.

However, with the use of our fast re-encryption method described in section V-B, the total number of exponentiations, $\mathcal{E}$ can be greatly reduced, down to $3S\cdot s\log_2 q + q + 16S\cdot s$. Given the cost of modular exponentiation (many orders of magnitude higher than modular multiplication), this reduction of the constant factor in q directly translates to a much smaller running time, in particular for cases where the number of explanatory variables is low ($s \ll q$).

V. IMPLEMENTATION OPTIMISATIONS

A. Parallelisation

The protocol can easily be run in a number of parallel processes: each sample is treatable independently of the others, meaning that the total computation time can theoretically be divided by up to S, given enough CPUs. In practice, we would want to avoid revealing the particular sign of the inner product from a single sample, but practical limitations (the large value of S and limited availability of CPUs) would always guarantee that each process would still handle a sufficiently large number of samples at a time to prevent such tying.

B. Fast Re-Encryption

Because modular exponentiation is costly and our protocol uses a large number of re-encryptions (which require one modular exponentiation each time), we designed a method that uses a finite set of pre-generated exponentiated random values to perform fast re-encryption (FRE) in the Paillier cryptosystem.

For judiciously chosen values of k and z (see security proof below), we can replace all calls to algorithm 2 in the main protocol by calls to algorithm 5 and save a large amount of computation without loss of security. Although this optimisation does not affect the asymptotic computational complexity of our protocol, it offers much better amortised cost, as can be seen from the computational cost analysis in section IV-D3 and the implementation test in section VII, in particular when the number of explanatory variables to be tested, s, is small compared to the number of samples used, S (typically $S = 10^3$ or $S = 10^4$, while s is in the order of 10^2).

As can be easily seen, after the z modular exponentiation in the set-up (executed once for the entire life of the cryptosystem's keys), the protocol only requires k multiplications to re-encrypt each value.

Proof of correctness is trivial if we note that this algorithm is equivalent to k executions of algorithm 2.

For security, we first remark that, aside from the general proof outline below, an adversary who could hypothetically tie re-encrypted values across all samples in our protocol would only be able to infer statistical information on the distribution of $\mathbf{x_2}$ across $\mathbf{y}$ (values $q_1, \ldots, q_m$ in section II), but no information about the actual individual values of elements of $\mathbf{y}$ or $\mathbf{y^{(i)}}$, as the semantic security of their initial encryption is guaranteed and their values are not used at any point during shuffling.

Algorithm 5 Fast Re-Encryption Method in Paillier Cryptosystem

parameters: Choose k and z such that they satisfy the security requirements (e.g. $k = 20, z = 1024$).
one-time setup: Generate z random values $r'_i \in \mathbb{Z}_N$ and store the modular exponentiated values: $[r_1, \ldots, r_z]$ where $r_i = {r'}_i^N \mod N$.
procedure FASTREENCRYPT(x)
input: Encrypted value $x = E(p)$
output: Encrypted value $y = E(p)$ such that x and y are statistically unrelated
    Pick k random elements of Z with replacement: $z_1, ..., z_k$
    $y \leftarrow x$
    for all z_i **do**
        $y \leftarrow y \cdot z_i \mod N$
    end for
    Return y
end procedure

For the sake of brevity, we only give the outline for an informal proof of security of this protocol, based on the security of algorithm 2:

Considering the scenario of an IND-CPA game where the adversary has so far received n ciphertexts of the same plaintext, generated by re-encryption using algorithm 5, we study the difficulty of telling an $(n+1)^{th}$ encryption apart from an unrelated ciphertext with probability significantly higher than chance.

With the same notations as algorithm 5, we note $R_i = r_{i_1} r_{i_2} \ldots r_{i_k}$ the blinding factor used during the i^{th} FRE and assume a worst-case scenario where the adversary learns each R_i values separately. Because of the negligibly small chance for accidental collision and the construction rules for R_i, distinguishing the $(n+1)^{th}$ re-encryption from another encryption with better probability than chance would require finding a combination of, possibly non-unique, R_i ($i \leq n$), in which R_{n+1} can be expressed. That is, finding two unordered collections S_1 and S_2 of (possibly non-unique) values between 1 and n, such that:

$$R_{n+1} = \frac{\prod_{i \in S_1} R_i}{\prod_{i \in S_2} R_i} \tag{4}$$

The security of the protocol therefore relies on the difficulty of finding the solution to problem 4, when it exists, which cannot be done by the adversary in polynomial time.

C. Early-Stopping

In the particular case where Bob can (or must) learn information on the final p-value attached to each covariate $\mathbf{x_{1i}}$, we can send samples in small increments (large enough to provide enough uncertainty about the value of $t_1^{(i)} - \hat{t}_1$ for a specific sample i) and remove an explanatory variable x_{1k} from the protocol as soon as the p-value is guaranteed to exceed the chosen significance threshold (this reveals how many samples were needed to establish significance, which in turns leaks critical information on the significance of the p-value). Because in practice, we expect far more non-significant than significant p-values, this quickly reduces the portion of the s explanatory variables that must be treated with each batch of samples.

VI. Comparison to Other Methods

To the best of the authors' knowledge, the present work is the first proof and implementation of a secure protocol to directly perform statistical testing on a logistic regression model (with data shared vertically between parties). Although many protocols for privacy-preserving GWAS already exist (such as secure computation of χ^2-statistics based on contingency tables [27]), they do not account for covariates that might be possible confounding factor in the model (e.g. important clinical data, such as smoking status) and affect the patient's outcome.

Fienberg *et al.* proposed a secure protocol to learn the parameters of a logistic regression model [28], [29], [2]. Once obtained, these parameters could be used at a relatively small cost to securely compute a p-value. As such, Fienberg's protocol for secure logistic regression (hereafter referred to as *FSLR*) is the closest alternative we know of that can perform such a test. We therefore used it as a baseline for comparison with our method.

Fienberg *et al.* do not provide a direct analysis of the complexity in homomorphic operations, but estimate the number of homomorphic operations needed by their protocol which lets us estimate a lower bound for the complexity in modular exponentiations in $\mathbb{Z}_N$ (we ignore the comparatively much cheaper multiplications in $\mathbb{Z}_N$):

Using the same notations as this paper, FSLR requires $\mathcal{O}(qd^2 + d^3 \log d)$ execution of their oblivious multiplication protocol and $\mathcal{O}(qL')$ executions of the GT comparison protocol described in [22], where d is the number of features used for the logistic regression and L' is the sampling size for their approximation of the sigmoid function. For our use, both d and L' can be seen as constants. Furthermore, we know that the GT comparison protocol they use requires $\mathcal{O}(\log N)$ encryptions and homomorphic operations. A very loose lower bound on the computational complexity of their protocol (in numbers of exponentiations in $\mathbb{Z}_N$) would therefore be: $\mathcal{O}(q + q \log N)$.

Additionally, each iteration of their algorithm has a communication round complexity in: $\mathcal{O}(L'P^2(qd^2 + d^3 \log d) + L'q) \in \mathcal{O}(q)$ (P being the total number of parties, here constant and equal to 2).

TABLE I
ASYMPTOTIC COMPLEXITIES OF PPELR COMPARED TO FSLR

	FSLR	PPELR
Round Complexity	$\mathcal{O}(sq)$	$\mathcal{O}(1)$
Computational Complexity in $\mathcal{E}$	$\mathcal{O}(sq + sq \log N)$	$\mathcal{O}(s \log q + q)$

From the summary in table I, we can see that while the asymptotic computational complexities of the two algorithms

are comparable, our approach requires a great deal less communication rounds between the two parties, which would give it a significant advantage in real-life applications where network latency might be an issue (the product of s and q would easily be in the order of 10^6).

Furthermore, we can note that, while it is difficult to evaluate the exact number of operations needed by FSLR, the constant terms are extremely high (L' needs to be at least 500 or 1000 to get stable results), whereas ours are fairly small, in particular when using the FASTREENCRYPT method described in subsection V-B. This is clearly illustrated by the difference in execution time between implementations of the two protocols (see section VII).

VII. IMPLEMENTATION AND TESTING

We implemented our protocol in C++ using the publicly-available Privacy-Preserving Protocol Framework[1] with 1024-bit encryption in the Paillier cryptosystem. We made and tested two versions of the algorithm: *PPELR* uses the regular REENCRYPT procedure described in subsection IV-A1, while *PPELR*$^+$ implements the optimised FASTREENCRYPT procedure from section V-B instead.

For comparison, we used a Java implementation of the FSLR protocol (slightly modified to use a polynomial approximation instead of the histogram approximation originally suggested by Fienberg *et al.*), also relying on Paillier encryption with 1024 bits.

We used experimental data collected from a cohort study on genetic variants linked to chronic kidney disease [30]. The experiment gathered clinical and Single-Nucleotide Polymorphism (SNP) data for $q = 4257$ patients that were also clinically assessed for high blood pressure (hypertension): $\mathbf{y} \in \mathbb{B}^s$. In the clinical data, we selected smoking status, a factor widely known to interfere with blood pressure [31], as a confounding variable ($\mathbf{x_2}$). The experiment measured up to 345 SNPs, for a total of 690 boolean variables ($\mathbf{x_{1i}}$) accounting for both full dominant and full recessive model of each SNP's minor allele.

Using high blood pressure and smoking status information as Alice's input, we computed p-values for $s = 1, 3, 10, 50, 100$ randomly chosen SNP variables (as Bob's input). For each value of s, we averaged the running times of 10 successive runs of the program. The sampling parameter S, was set to 1000, a value that gives us enough precision to safely rule out non-significant SNPs.

When using the FSLR protocol to compute p-values it is necessary to train a separate model for each SNP, meaning that the computing time for s SNP variables is very exactly s times the computing time for a single variable. Additionally FSLR requires choosing the maximum number of iterations over which to refine the regression parameters. Our testing showed that 5 iterations gave a somewhat satisfying convergence for the log-likelihood of the estimated parameters (using a non-oblivious solver to provide the optimal value).

[1] https://github.com/david-duverle/pppf

Of the 100 first SNP variables tested with our protocol, 6 were shown to be associated with the outcome at a significance level of 5%, of which one was still significant ($p = 0.0045$) after Bonferroni adjustment for multiple hypotheses testing ($k = 100$).

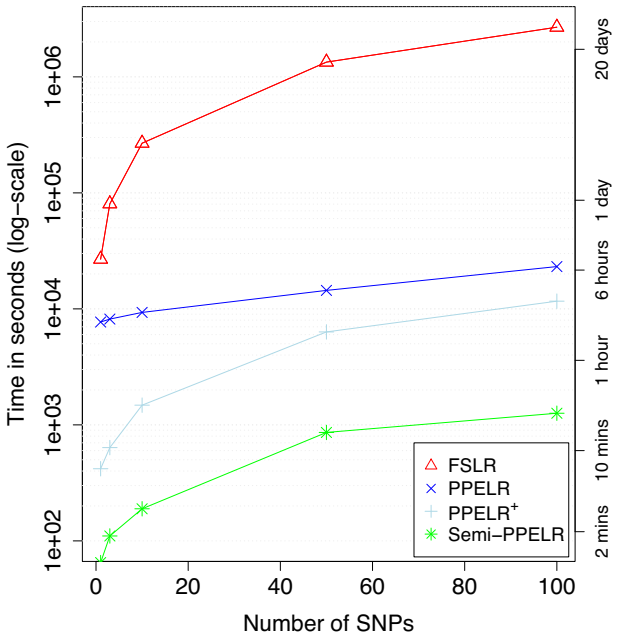

Fig. 2. Comparison of time performances for our protocol against FSLR: the large difference in time complexity can be readily observed in the much steeper growth (measured on a log-scale) for FSLR compared to our methods.

All computation times were measured on a single Intel Core i7 2.7Ghz CPU. The average computation times of each method for the different values of s are summarised in figure VII. As we can observe, not only is PPELR's computational time growing much slower in the number of SNPs handled, but the constant factor is many orders of magnitude lower: obtaining p-values for 100 SNP variables would take more than 27 days with FSLR, whereas our method can give such a result in 3.23 hours (6.43, if not using FRE). We can also see that our FRE technique practically erases the constant computational cost (mainly caused by the initial generation of re-encrypted randomised samples), leading to a significant time gain, especially for smaller numbers of SNPs: it takes PPELR$^+$ 7 minutes (418s) to test a single SNP, versus more than 2 hours (7,702s) for PPELR.

Additionally, we implemented and tested the 'early-stopping' variant of our protocol described in section V-C (noted *Semi-PPELR*), where both parties *de facto* learn the p-values for each variable (but all input still remains private). As an early-stop threshold, we used the (very conservative) significance threshold of 0.01, without further accounting for multiple hypotheses testing. Furthermore, we only eliminated variables that had already exceeded the final p-value ratio, rather than run a statistical estimate based on the ongoing

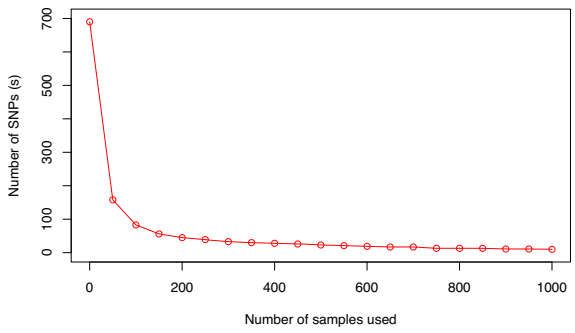

Fig. 3. Reduction of the number of SNPs (s) still tested as more samples are returned. From an initial number of 690 SNP variables, only 45 are still considered potentially significant (at significance level 0.01) after 200 samples have been run (13, after 700 samples).

number of samples tested (which would presumably prune an even larger number of variables during the earlier steps).

As can be seen in figure VII, most of the SNP variables are very quickly removed from consideration by the algorithm, explaining the much lower computational times in figure VII.

Total running time on the full set of SNP variables ($s = 790$) with Semi-PPELR was less than 2 hours and a half (8470s).

VIII. Improvements and Future Directions

Among the more immediate improvements to our protocol, we plan to test the use of more efficient additive cryptosystems, such as Damgård-Jurik cryptosystem's generalisation of Paillier's [32], which would be likely to decrease the constant factor cost of the computations.

We are also considering the use of so-called "somewhat-fully" homomorphic encryption, such as provided by Ring-Learning With Error (RLWE) encryption [33], which could help provide a more efficient way to run our comparisons. The additional use of RLWE's plaintext packing techniques [34] could prove extremely beneficial to our problem setting, which typically involves large arrays of binary values.

In terms of the algorithm itself, we are planning to extend its capabilities, by formalising the use of multiple covariates, using Markov chain Monte Carlo methods [17], as well as studying variants of the oblivious set-up where the data might be shared horizontally, rather than vertically (multiple institutions sharing the same variables over different populations).

IX. Conclusion

In this work we have demonstrated the first use of a sampling-based method to perform exact statistical testing on logistic regression models in a privacy-preserving context. As we have shown, our method is not only as secure and accurate as existing methods, but provides a marked improvement in both theoretical, and practical, performances over the closest existing equivalent.

In its current implementation, our protocol can be used to provide exhaustive statistical testing of large data sets (tested on a data set of over 4200 instances with 690 explanatory variables) while accounting for confounding factors, and produce results in a matter of minutes, with further room for improvement through parallelisation. Such performances finally make covariate-aware GWAS analysis in privacy-preserving context, a practical possibility with real-life genomic data.

A fully functional and reusable implementation of our protocol is available online[2].

Acknowledgment

This work is supported by JST CREST program "Advanced Core Technologies for Big Data Integration". The authors would like to thank Shuang Wu, from Tsukuba university, Kota Matsui and Ichiro Takeuchi, from Nagoya Institute of Technology, for providing their implementation of the FSLR protocol based on the work of Fienberg *et al.*, and Noboru Kunihiro, from the University of Tokyo, for his helpful suggestions.

References

[1] G. H. Fernald, E. Capriotti, R. Daneshjou, K. J. Karczewski, and R. B. Altman, "Bioinformatics challenges for personalized medicine," *Bioinformatics*, vol. 27, no. 13, pp. 1741–1748, 2011.

[2] Y. Nardi, S. E. Fienberg, and R. J. Hall, "Achieving both valid and secure logistic regression analysis on aggregated data from different private sources," *Journal of Privacy and Confidentiality*, vol. 4, no. 1, p. 9, 2012.

[3] J. Johnson, B. Burkley, T. Langaee, M. Clare-Salzler, T. Klein, and R. Altman, "Implementing personalized medicine: development of a cost-effective customized pharmacogenetics genotyping array," *Clinical Pharmacology & Therapeutics*, vol. 92, no. 4, pp. 437–439, 2012.

[4] N. Siva, "1000 genomes project," *Nature biotechnology*, vol. 26, no. 3, pp. 256–256, 2008.

[5] S. Sankararaman, G. Obozinski, M. I. Jordan, and E. Halperin, "Genomic privacy and limits of individual detection in a pool," *Nature genetics*, vol. 41, no. 9, pp. 965–967, 2009.

[6] C. Cachin, I. Keidar, and A. Shraer, "Trusting the cloud," *Acm Sigact News*, vol. 40, no. 2, pp. 81–86, 2009.

[7] ——, "Fail-aware untrusted storage," *SIAM Journal on Computing*, vol. 40, no. 2, pp. 493–533, 2011.

[8] R. M. Cantor, K. Lange, and J. S. Sinsheimer, "Prioritizing gwas results: a review of statistical methods and recommendations for their application," *The American Journal of Human Genetics*, vol. 86, no. 1, pp. 6–22, 2010.

[9] R. Sokal and F. Rohlf, *Biometry: 3rd edition*. Freeman, 1995.

[10] C. R. Mehta and N. R. Patel, "Exact logistic regression: theory and examples," *Statistics in medicine*, vol. 14, no. 19, pp. 2143–2160, 1995.

[11] K. Hirji, *Exact Analysis of Discrete Data*. Taylor and Francis, 2006.

[12] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. Ferreira, D. Bender, J. Maller, P. Sklar, P. I. De Bakker, M. J. Daly *et al.*, "Plink: a tool set for whole-genome association and population-based linkage analyses," *American Journal of Human Genetics*, vol. 81, no. 3, pp. 559–575, 2007.

[13] S. Menard, *Applied logistic regression analysis*. Sage, 2002, vol. 106.

[14] D. Zamar, B. McNeney, and J. Graham, "elrm: Software implementing exact-like inference for logistic regression models," *J. Statist. Software*, vol. 21, pp. 1–18, 2007.

[15] P. Diaconis and B. Sturmfels, "Algebraic algorithms for sampling from conditional distributions," *Annals of Statistics*, vol. 26, no. 1, pp. 363–397, 1998.

[16] V. Karwa and A. Slavkovic, "Conditional inference given partial information in contingency tables using Markov bases," *WIREs Comput. Stat.*, vol. 5, pp. 207–218, 2013.

[17] C. R. Mehta, N. R. Patel, and P. Senchaudhuri, "Efficient monte carlo methods for conditional logistic regression," *Journal of the American Statistical Association*, vol. 95, no. 449, pp. 99–108, 2000.

[18] D. A. Pierce and D. Peters, "Improving on exact tests by approximate conditioning," *Biometrika*, vol. 86, no. 2, pp. 265–277, 1999.

[2]https://github.com/david-duverle/ppelr

[19] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptologyEUROCRYPT99*. Springer, 1999, pp. 223–238.
[20] P. Y. Ryan, "Prêt à voter with paillier encryption," *Mathematical and Computer Modelling*, vol. 48, no. 9, pp. 1646–1662, 2008.
[21] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *Information Security and Cryptology–ICISC 2004*. Springer, 2005, pp. 104–120.
[22] I. F. Blake and V. Kolesnikov, "One-round secure comparison of integers," *Journal of Mathematical Cryptology*, vol. 3, no. 1, pp. 37–68, 2009.
[23] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *Public Key Cryptography–PKC 2007*. Springer, 2007, pp. 343–360.
[24] A. C. Yao, "Protocols for secure computations," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 1982, pp. 160–164.
[25] I. Ioannidis and A. Grama, "An efficient protocol for yao's millionaires' problem," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. IEEE, 2003, pp. 6–pp.
[26] S.-D. Li, Y.-Q. Dai, and Q.-Y. You, "Efficient solution to yao's millionaires' problem." *Dianzi Xuebao(Acta Electronica Sinica)*, vol. 33, no. 5, pp. 769–773, 2005.
[27] S. E. Fienberg, A. Slavkovic, and C. Uhler, "Privacy preserving gwas data sharing," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 628–635.
[28] S. E. Fienberg, Y. Nardi, and A. B. Slavković, "Valid statistical analysis for logistic regression with multiple sources," in *Protecting Persons While Protecting the People*. Springer, 2009, pp. 82–94.
[29] R. Hall, S. E. Fienberg, and Y. Nardi, "Secure multiple linear regression based on homomorphic encryption," *Journal of Official Statistics*, vol. 27, no. 4, p. 669, 2011.
[30] T. Yoshida, K. Kato, T. Fujimaki, K. Yokoi, M. Oguri, S. Watanabe, N. Metoki, H. Yoshida, K. Satoh, Y. Aoyagi *et al.*, "Association of genetic variants with chronic kidney disease in japanese individuals," *Clinical Journal of the American Society of Nephrology*, vol. 4, no. 5, pp. 883–890, 2009.
[31] M. S. Green, E. Jucha, and Y. Luz, "Blood pressure in smokers and nonsmokers: epidemiologic findings," *American heart journal*, vol. 111, no. 5, pp. 932–940, 1986.
[32] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Public Key Cryptography*. Springer, 2001, pp. 119–136.
[33] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 505–524.
[34] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *Public-Key Cryptography–PKC 2013*. Springer, 2013, pp. 1–13.