Privacy-Preserving Verification of Clinical Research

Eleftheria Makri[1] Maarten H. Everts[1,3] Sebastiaan de Hoogh[2]
Andreas Peter[1] Harm op den Akker[1,4] Pieter H. Hartel[1,3]
Willem Jonker[1]

[1] University of Twente
[2] Eindhoven University of Technology
[3] TNO, Netherlands Organisation for Applied Scientific Research
[4] Roessing Research and Development

Abstract: We treat the problem of privacy-preserving statistics verification in clinical research. We show that given aggregated results from statistical calculations, we can verify their correctness efficiently, without revealing any of the private inputs used for the calculation. Our construction is based on the primitive of Secure Multi-Party Computation from Shamir's Secret Sharing. Basically, our setting involves three parties: a *hospital*, which owns the private inputs, a *clinical researcher*, who lawfully processes the sensitive data to produce an aggregated statistical result, and a third party (usually several *verifiers*) assigned to verify this result for reliability and transparency reasons. Our solution guarantees that these verifiers only learn about the aggregated results (and what can be inferred from those about the underlying private data) and nothing more. By taking advantage of the particular scenario at hand (where certain intermediate results, e.g., the mean over the dataset, are available in the clear) and utilizing secret sharing primitives, our approach turns out to be practically efficient, which we underpin by performing several experiments on *real* patient data. Our results show that the privacy-preserving verification of the most commonly used statistical operations in clinical research presents itself as an important use case, where the concept of secure multi-party computation becomes employable in practice.

1 Introduction

Statistical analysis of experimental data is the cornerstone in many research areas. However, human error and fraud are common threats to the integrity of the statistical results [Fan09, Ens, Mis, RBG+00]. In addition, verification of such statistical results cannot be applied in a straight-forward manner, since in many cases the underlying data has to remain confidential. To address this problem, we propose a privacy-preserving verification procedure that allows a number of semi-honest verifiers to ascertain that statistical calculations are consistent with the confidential data that they are supposedly based on, without learning about this underlying confidential data.

In medical research, it is common practice to give clinical researchers access to raw patient data. This is necessary for researchers to determine the appropriate statistical analysis

method for the specific dataset in question [Fie09, pp. 822]. Patient privacy in that context is preserved by the researchers themselves, who are bound by confidentiality agreements. Currently, only the most prestigious medical scientific journals like Thorax [Tho] perform statistics verification on the clinical research results, prior to their publication. This is a labor intensive task, which has to be performed by expert statisticians. In addition, there is a trade-off between patient privacy and thoroughness of the verification procedure. On the one hand, if the results are only partially checked, thoroughness is sacrificed to preserve some privacy. On the other hand, if all the results are thoroughly checked, then patient privacy will be completely compromised. Note that current anonymization techniques have been shown insecure, since anonymized data can be de-anonymized [Swe02]. Thus, disclosure of (possibly anonymized) patient information should not be allowed; not even to medical journals for verification.

Although hospitals have the confidential patient data used for the statistical analysis available in the clear, they currently do not consider the verification of the statistics. This is because 1) hospitals wish to avoid the additional workload brought by the verification; 2) clinical researchers are usually employed by the same hospital that provides them with the data, where a conflict of interests might arise; 3) on-site verification does not scale, since it is not possible to verify the results accruing from datasets of different hospitals (i.e., in the case of a multi-center clinical research). In contrast, medical journals are interested in the correctness of the results that they publish. Therefore, we propose that journals outsource the verification of statistics to an independent group of servers, called the *verifiers*, in a privacy-preserving manner. In this setting, the architecture that we propose is depicted in Fig. 1. Our approach can be fully automated and does not require additional manpower to be employed. This may well serve as a motivation for all (medical) journals to implement this paradigm and integrate verification into their pre-publication process.

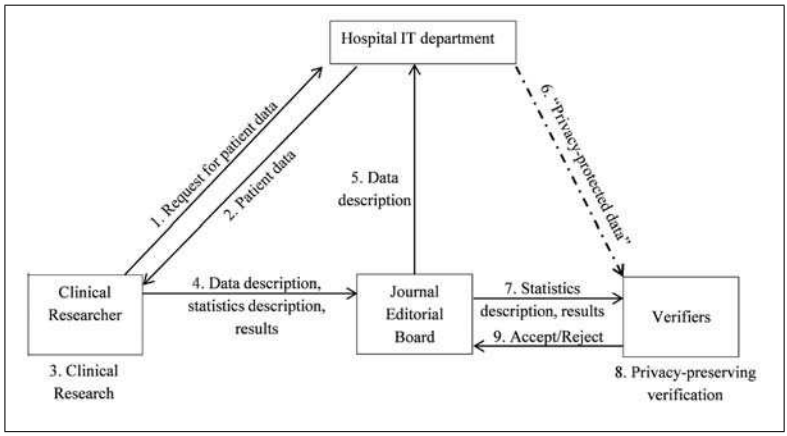

Figure 1: Privacy-preserving architecture for the verification of clinical research

Concretely, we make the following contributions:

Enhance Privacy-Awareness in the Verification of Clinical Research. Patient data is confidential and is only to be disclosed to (trusted) experts conducting the clinical research

with the explicit patients' informed consent. External parties, such as medical journals, should not receive patient confidential information, even if it has been anonymized. We point out this important issue and propose a best practice framework (Fig. 1).

Enable Privacy-Preserving Verification of Clinical Research. As the verification of clinical research at the hospital site is unsuitable (for the mentioned reasons, such as conflict of interests), we propose a mechanism that allows for the outsourcing of this verification to several semi-honest *verifiers* without compromising the confidentiality of the patients' data. Our approach is based on secure multi-party computation from Shamir's secret sharing and is proven secure in the semi-honest model. We base our protocols on secret sharing, because of the storage and computational efficiency it provides. For Shamir's secret sharing to be secure, while allowing the evaluation of polynomial functions over the shares, we need at least three computing parties. Hence, we assume several (a minimum of three) *independent*, non-colluding verifiers performing the verification task. The contractual relationship between the journal and the verifiers, prevents the verifiers from cheating. Thus, our application scenario allows us to work in the semi-honest model, where the parties are assumed to follow the protocol correctly, but they are able to record the protocol transcript, in order to infer private information.

Demonstrate the Practicality of our Approach with Real Patient Data. We develop a set of privacy-preserving algorithms, which allows the verification of the most commonly used statistical operations in clinical research [Md09, OS08, ZBT07]: mean, variance, Student's t-test, Welch's t-test, ANOVA (F-test), simple linear regression, chi-squared test, Fisher's exact test, and McNemar's test. We test this representative set of algorithms on a *real* medical dataset, produced in the context of a tele-treatment medical research [AodJH10], and show their efficiency. The dataset consists of 2370 activity feedback messages produced for 85 patients, participating in the research. We also doubled and tripled this dataset to show the scalability of our solution.

The execution times of our verification algorithms on this dataset range from 43.5 ms in the fastest case (a verification of the mean age of 84 patients) to 884.6 ms in the slowest case (a verification of simple linear regression on 6828 messages). We refer to Section 5.2 for details. Our execution times, in combination with the fact that we use real data, substantiates the practicality of our approach.

Our solution can detect any error in the computation of the statistics, but it cannot detect logical errors (i.e., errors in the selection of the appropriate statistical model that fits the data best). Thus, our solution is not meant to substitute, but rather to enhance the peer-reviewing process of medical journals. Currently, the statistics that our solution can treat, are limited to the ones meant for normally distributed data. Dealing with statistics meant for not normally distributed data requires highly efficient ranking of the inputs, which is a challenging task in the secret shared domain. We leave it as an interesting future work.

The rest of the paper is organized as follows: In Section 2, we discuss related work and frame our contribution within it. In Section 3, we give the preliminaries on the secret sharing and the multi-party computation techniques that we utilize. In Section 4, we discuss the proposed verification algorithms and the statistics to be verified. Section 5 deals with the security and performance analysis of our algorithms and Section 6 concludes the paper.

2 Related Work

A lot of work has been done to solve the problem of privacy-preserving statistical analysis [DA01, DHC04, SCD+08, DN04, HR10, DF97, KLSR09]. In this setting, aggregated statistical results are being computed among several parties, while the inputs for the calculation of these results remain private. In contrast to our work, their attention is focused on secure *computation* of certain statistics, where each party that is involved in the computation provides its own private data. We deal with the *verification* of statistics, for which the verifiers do not contribute their own private data. In that context, privacy concerns data provided by a party, which is not involved in the computation.

Another related topic in our setting is *verifiable computation* [GGP10, PRV12, AIK10, BGV11, PST13]. Verifiable computation allows a party (or set of parties) to outsource the computation of certain functions to untrusted external parties, while maintaining *verifiable* results. However, despite their lack of efficiency, all existing constructions in this area are not applicable in our setting, as the verification is not meant to be privacy-preserving. The security definitions of these works guarantee that the untrusted computing parties cannot cheat in the computation (i.e., a false result will not pass the verification procedure).

Another recently emerged approach towards addressing the problem of privacy-preserving verification is computing on authenticated data [ABC+12], which can be accomplished using homomorphic signatures. The concept of homomorphic signatures emerged from [JMSW02], where the first example application scenarios and definitions were given. Recently, the application of homomorphic signatures was extended from treating only set operations, to computation of functions on the signed data [BF11b, BF11a]. In [BF11b], Boneh and Freeman propose a homomorphic signature scheme that allows linear functions to be performed on the signed data, while anyone can produce a signature on the result of the function. The scheme is weakly context hiding, meaning that it protects the privacy of the signed data, but not hiding the fact that the function was executed. Although the aforementioned requirement fits well in our scenario, the scheme can only treat linear functions, which is insufficient for our purposes, as we need to compute additions and multiplications with non-constants. Building upon their previous work [BF11b], Boneh and Freeman [BF11a] propose a homomorphic signature scheme for polynomial functions. However, the enhanced functionality of the latter scheme [BF11a] comes at the cost of completely loosing the property of context hiding. This makes it inadequate for our scenario, as it provides no privacy of the underlying inputs at all.

Finally, Thompson et al.'s work [THH+09] deals with computing and verifying aggregate queries on (private) outsourced databases. They look at a setting, where a data owner outsources his private database to third-party service providers, who can later answer aggregate queries of external users or the data owner himself. Although their work is based on similar building blocks as ours, their setting is slightly different as it focuses on the computation of a very limited class of statistics and not on their verification. Exploiting the fact that we only perform verification (instead of computation from scratch) allows us to achieve very high efficiency compared to their work, while being able to deal with more sophisticated statistics at the same time.

The efficiency and hence practicality of Multi-Party Computation has received a lot of attention in the last few years [BCD+09, BSMD10, BLW08]. Our solution, similarly to the work of Bogetoft et al. [BCD+09] is based on VIFF [VD], which we used to implement the proposed privacy-preserving verification algorithms.

3 Preliminaries

We recall Shamir's secret sharing [Sha79] and Secure Multi-Party Computation [GRR98], which form the main building blocks of our protocols.

3.1 Shamir's Secret Sharing

A secret sharing scheme is a protocol, where a special party called the *dealer*, wishes to share a secret value $s \in \mathbb{F}_q$ (i.e., a secret value in a finite field of order q), among the set of protocol participants. A subset of the protocol participants, called the *qualified set*, can reconstruct the original secret value. Shamir's secret sharing scheme [Sha79] consists of three subprotocols: the share generation, secret sharing and secret reconstruction. During the share generation, the *dealer* of the protocol chooses a random polynomial (over some finite field $\mathbb{F}_q$) $p(x) = \alpha_t x^\tau + ... + \alpha_1 x + s$ of degree τ, where $\tau+1$ is the number of players in the qualified set and s is the secret to be shared. Then, he evaluates the polynomial for each player as follows: $[s]_i := p(i)$, where $[s]_i$ denotes the share of s of the i^{th} player P_i. For secret sharing all the shares have to be distributed to the players via secure channels. Then, to reconstruct the secret, anyone who possesses at least $\tau + 1$ shares can interpolate the polynomial (e.g., by Lagrange interpolation) and reconstruct the original secret s.

3.2 Secure Multi-Party Computation based on Secret Sharing

Due to the additively homomorphic property of Shamir's secret sharing scheme, addition and subtraction of Shamir's shares can be performed directly on the shares (locally by each player), without requiring any interaction. This means that after having added/subtracted the shares, we can reconstruct the resulting secret, and the result of the reconstruction will be the correct result of the summation/subtraction. The same holds for addition, subtraction and multiplication with public constants on Shamir's shares. Computing the sum of the shares of a secret vector's elements and then opening this result is commonly used for the construction of our privacy-preserving verification algorithms; we will denote the protocol for the aforementioned computation SumPub(.). The input for this protocol is the shares of a secret shared vector of integer values, and the output is the result of the addition of the vector elements (in the clear).

To multiply with shares, we need an interactive protocol to be executed among the players. In [GRR98], an interactive protocol for multiplication of Shamir's shares is proposed,

which completes its execution in one communication round. This protocol reduces the degree of the underlying polynomial (which has been doubled due to the multiplication) and restores its randomness, requiring one interactive operation for both. Our verification algorithms heavily depend on the computation of inner products in the secret shared domain. For the computation of inner products, we use a generalized version of the multiplication protocol [GRR98], also discussed in [CHd10], that comes at the same round communication cost as the multiplication (i.e., one communication round). This protocol (called InnerPub(.)) performs all the necessary operations in the secret shared domain and then reconstructs the result of the inner product. This is done by first using the PRZS(.) protocol (Pseudorandom Zero-Sharing), proposed in [CDI05], to distribute the shares of a 0 to the protocol participants; those shares are later added to the share of the multiplication result, to restore its randomness. The summation of the products and the reconstruction of the result together, require only one communication round, in which the shares are distributed, the summation is computed and the result is reconstructed. The input for the InnerPub(.) protocol is the shares of two secret shared vectors of integer values, and the output is the result of computing the inner product on these vectors (in the clear).

4 Privacy-Preserving Statistics Verification

In this section, we describe our protocols for privacy-preserving verification of statistics, while dealing with each statistical test separately. Prior to the execution of each protocol, we assume that the original data used in the calculation has been properly secret shared among the verifiers (by the hospital), according to Shamir's Secret Sharing scheme [Sha79]. The main ingredients for our verification protocols, which act in the secret shared domain, are the InnerPub(.) and SumPub(.) protocols discussed in the Preliminaries.

Our approach takes advantage of the fact that all statistical results to be verified are public. Thus, we can let the verifiers recompute the statistics in the secret shared domain, reconstruct the shared results, and compare with the statistics to be verified in the clear. In most cases, we can also take advantage of intermediate results, since they are public. Note that although our building blocks work on integers, we can also handle fixed point numbers, by scaling them up to the desired precision and then treat them as integers.

4.1 Privacy-Preserving Mean Verification

The mean (or average) is one of the simplest statistics and is computed as

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}, \tag{1}$$

where N is the size of the sample (i.e., the number of individual subjects to be analyzed) and x_i is the variable concerning the i^{th} subject.

Recall that we assume that all x_i ($i = 1, .., N$) have been secret shared among the verifiers. We compute the value $\bar{x}$ in a privacy-preserving manner, by letting the verifiers run SumPub(.). This yields the result $\sum_{i=1}^{N} x_i$. The opening of this result does not constitute a privacy violation, since both the mean value to be verified and the size of the sample are public values, and the result of the summation can be computed by those two values.

By taking advantage of the fact that the number of subjects N is public, we do not perform the division by N in the secret shared domain. Instead, we multiply the mean value (that is to be verified) by N and compare it with the sum $\sum_{i=1}^{N} x_i$ that the verifiers computed (for efficiency reasons). The pseudo-code is given in Algorithm 1 (Appendix A).

4.2 Privacy-Preserving Variance Verification

The variance is a measure, assessing how far the real observations are from the expected value (i.e., the mean value discussed in the previous subsection) and is computed as

$$S^2 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}. \tag{2}$$

The variance only makes sense if published together with the mean value. Thus, the mean value corresponding to the variance will also be public. Our variance verification algorithm first verifies the aforementioned mean value, since it cannot be safely used, unless verified. Next, the verifiers compute all the N subtractions locally on the shares of x_i and then interactively compute an inner product (with public result) on the results of these subtractions. This is performed by invoking the InnerPub(.) protocol, discussed earlier. Similarly to the mean verification protocol, we avoid the division by $N - 1$ in the secret shared domain and perform a multiplication of the received variance (to be verified) by $N - 1$, instead. Then, the verifiers check the consistency of the latter product, with the inner product computed earlier and if any of them fails to find a match, the verification fails. Details are given in Algorithm 2 (Appendix A).

4.3 Privacy-Preserving Student's t-test Verification

The Student's t-test is one of the most frequently used statistical tests to assess the significance of a statistical hypothesis and is calculated as

$$t_{student} = \frac{\frac{\sum x_i}{N} - \frac{\sum y_i}{N}}{\sqrt{\frac{\frac{\sum (x_i - \bar{x})^2}{N-1} + \frac{\sum (y_i - \bar{y})^2}{N-1}}{N}}} = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{S_x^2 + S_y^2}{N}}}, \tag{3}$$

where S_x^2 (resp. S_y^2) is the variance of variable x (resp. y).

The Student's t-test depends on: the variables (x and y) on which it is computed, which we treat as the private inputs that are secret shared among the verifiers; the mean and variances

corresponding to those variables; and the number of subjects N. In the context of clinical research, the variables x and y can be blood pressures of N patients, after being treated with medication X and Y respectively, and the hypothesis could concern the effectiveness of these two medications in reducing the blood pressure. As we will use the public means and variances for the verification of the t-value, we first need to verify them. After having verified the public mean values and the corresponding variances, the verifiers act only in the plaintext domain to evaluate equation (3), and check the consistency of this result with the one received for verification. For further details see Algorithm 3 (Appendix A).

4.4 Privacy-Preserving Welch's t-test Verification

The Welch's t-test is a variation of the Student's t-test, for the cases where the two groups x, y under consideration, have different variances and sizes. The formula for calculating Welch's t-test is given by

$$t_{welch} = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{S_x^2}{N_x} + \frac{S_y^2}{N_y}}}, \tag{4}$$

where N_x (resp. N_y) is the size of group x (resp. y).

After having verified the mean values and the corresponding variances, there is no other calculation or verification to be performed in the secret shared domain. Having verified the means and variances for equation (4), and given that N_x, N_y are public, the verifiers evaluate equation (4). To complete the verification, the verifiers compare the result of the aforementioned privacy-preserving evaluation with the t-value to be verified. The details of the verification algorithm for Welch's t-test are given in Algorithm 4 (Appendix A).

4.5 Privacy-Preserving F-test Verification

The F-test is one of the most commonly used tests as part of the Analysis Of Variance (ANOVA). ANOVA is used to determine the significance of a statistical hypothesis. It is used instead of a t-test when there are more than two groups for which the significance of the difference among them needs to be determined. We can compute the F-value as

$$F = (\sum_{i=1}^{K} \frac{N_i(\bar{x}_i - \bar{X})^2}{K-1}) / (\sum_{i=1,j=1}^{K,N_i} \frac{(x_{ij} - \bar{x}_i)^2}{G-K}), \tag{5}$$

where K is the number of groups under analysis, N_i is the size of the i^{th} group, $\bar{x}_i$ is the mean of the i^{th} group, $\bar{X}$ is the mean of all group means, x_{ij} is the j^{th} (private) value of group i, and G is the total size (i.e., the sum of all N_i). The privacy-preserving variant of this, works as follows: the verifiers compute and verify the means and variances of all groups. Then, they compute in the clear the overall mean $\bar{X}$ and the total size G. Given

the aforementioned intermediate results, the verifiers evaluate equation (5) and compare the result to the F-value to be verified. The detailed protocol is listed in Algorithm 5.

4.6 Privacy-Preserving Simple Linear Regression Verification

Simple linear regression is used to predict an outcome (or dependent) variable from one predictor (or explanatory) variable. Specifically for simple linear regression, what needs to be calculated is the coefficients of the straight line

$$y = \alpha x + \beta \tag{6}$$

where the coefficient β is given as $\beta = \frac{\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{N}(x_i-\bar{x})^2}$, and α is given as $\alpha = \bar{y} - \beta\bar{x}$. The main ingredients for the calculation of β are the mean values of each of the two groups. Thus, the verifiers begin with verifying the means. Observe also that the denominator of the fraction is the sum of squared errors, which can be computed by invoking the InnerPub(.) protocol with the appropriate arguments (as seen before in the verification of the variance). The numerator of the fraction is also an inner product that the verifiers compute. The result of this inner product is allowed to be revealed to the verifiers, because the coefficient β is part of the public result of linear regression and this value can be directly determined by multiplying β with the sum of squared errors over variable x, which is also public (as part of the variance). Having calculated all the aforementioned, the verifiers compute β and compare it with the received one. Next, they calculate α and proceed to its comparison with the α received for verification. After those two comparisons, the parameters of the straight line, accruing from the simple linear regression have been verified. Our detailed protocol for this test is listed in Algorithm 6 (Appendix A).

4.7 Privacy-Preserving Chi-Squared test Verification

The Chi-squared test [Pea00] and the two tests that follow in the next subsections, namely Fisher's exact test [Fis22] and McNemar's test [McN47], are all statistical tests meant to be used when the underlying data is categorical. This means that those tests examine the frequency distributions of observations in a group. The frequencies $observed_{ij}$ are recorded in a table, called the contingency table, where the row and column totals (i.e., the sum of each row's elements $RowTotal_i$ and column's elements $ColumnTotal_j$, respectively) are also recorded. The most well-known chi-squared test (and the one that we treat in this paper) is Pearson's χ^2-test [Pea00]. In particular, this test measures how well the experimental data fits in the chi-squared distribution. The formula for Pearson's χ^2-test is

$$\chi^2 = \sum \frac{(observed_{ij} - model_{ij})^2}{model_{ij}}, \tag{7}$$

where $model_{ij} = (RowTotal_i \cdot ColumnTotal_j)/N$.

To enable privacy-preserving verification of tests meant for categorical data, a preprocessing step is required. During the preprocessing phase, the raw data is encoded to its unary representation. The number of categories, in each dimension of the clinical research, defines the number of bits of each entry in the table of raw data. For example, if we were examining the effect of 3 different medications, the number of bits of each entry in the medication column would have been also 3. After having successfully preprocessed the data, the hospital secret shares this data bitwise among the verifiers. We require this preprocessing step, as it allows us to efficiently compute the frequencies in the contingency table, by only adding up the data, or computing their inner product column-wise.

For the verification of the χ^2-test, we need to verify the frequencies of the variables in a contingency table. This is common to all statistical tests meant for categorical data and we present this step in a separate algorithm (VrfContingency(.), Algorithm 7 in Appendix A), for reusability purposes. The contingency table verification is performed by calculating the inner product of each bitwise secret shared variable's column, with the second variable's corresponding column. This is what the verifiers compute in the secret shared domain and then check its consistency with the table received for verification. The total and the marginal totals do not require calculations in the secret shared domain to get verified, since they can be computed by adding up the (verified) frequencies. Having computed the total and marginal totals, and verified the frequencies in the contingency table, the verifiers evaluate equation (7) and compare the result to the value to be verified. Our detailed algorithm for privacy-preserving χ^2-test verification is given in Algorithm 8 (Appendix A).

4.8 Privacy-Preserving Fisher's exact test Verification

Fisher's exact test [Fis22] gives us the *exact* p-value determining whether the relationship between the variables of the model is significant. This is in contrast to the χ^2-test, which is an *approximation* of the significance. Fisher's exact test is used with small sample groups, while χ^2-test is more suitable for large ones. The formula for Fisher's exact test is

$$p = \frac{\prod (RowTotal_i! \cdot ColumnTotal_j!)}{\prod observed_{ij}! \cdot N!}. \tag{8}$$

Fisher's exact test is similar to the χ^2-test in terms of verification. This is because all that needs to be verified is the frequencies in the contingency table. The same preprocessing on the original data, as the one performed for the χ^2-test is required. The verifiers begin with the verification of the frequencies in the contingency table, and the computation of the total and the marginal totals (same as in χ^2-test). Then, they evaluate equation (8) and check the consistency of the result with the one to be verified. Our detailed algorithm for Fisher's exact test verification is given in Algorithm 9 (Appendix A).

4.9 Privacy-Preserving McNemar's test Verification

McNemar's test [McN47], similarly to Pearson's χ^2-test, gives us an approximation of the significance. This test, given by the following formula

$$\chi^2 = \frac{(observed_{1,2} - observed_{2,1})^2}{observed_{1,2} + observed_{2,1}}, \tag{9}$$

can be only applied to data recorded on a 2×2 contingency table. For McNemar's test, what we verify in the secret shared domain is the frequencies of the contingency table, as in Fisher's exact test and χ^2-test. Hence, the verifiers proceed as described in χ^2-test's verification. Then, they evaluate equation (9) and compare the result with the χ^2-value received for verification. Details are given in Algorithm 10 (Appendix A).

5 Security and Performance Analysis

Our setting lies in the semi-honest model, meaning that the verifiers are assumed to honestly follow the instructions mandated by the protocol, but they wish to learn as much information as possible about the private inputs of the *dealer* (i.e., the hospital). The verifiers are allowed to know all the public inputs given as arguments in the protocols (e.g., the sample group size N of the statistical operation to be verified) and all the public results that they compute. The verifiers are not allowed to learn anything more than the aforementioned, in addition to what can be inferred by the results. Hence, we need to protect the private inputs of the *dealer* and we do so by means of Shamir's Secret Sharing. This way we achieve information-theoretic security, as long as at least $\tau > \frac{n}{2}$ verifiers are honest and do not collude. We assume that there exists pairwise secure channels between the verifiers.

Our performance analysis is based on a proof of concept implementation that we designed to demonstrate the efficiency of our solution. We used real patient data for our experiments to show the applicability of our proposal in practical cases. The aforementioned data concerns patient compliance in a tele-treatment application, where the patients were carrying a monitoring system, measuring their activity and sending them back activity advice, in the form of feedback messages. This dataset consists of 2370 feedback messages of 85 patients that have been analyzed. For more information about the data and the tele-treatment application we refer the reader to [AodJH10].

5.1 Security Analysis

The security requirement that we wish to satisfy is to preserve the confidentiality of the private inputs of the *dealer*, while allowing a certain functionality of the verifiers, enabling the verification of the result of a predefined function. We deal with passive, static adversaries corrupting any minority of the verifiers. Security is modeled using the real vs. ideal

paradigm [Gol04, Section 7.2]. In the real world, the protocol participants execute the protocol interactively and there is an adversary $\mathcal{A}$, having access to all the private inputs and all the messages exchanged among the corrupted parties, as well as the public inputs of both the corrupted and the honest parties. In the ideal world, a protocol is assumed to be executed in the presence of a trusted party, which the protocol participants query and get the appropriate results, based on their predefined functionality. To prove security in the real vs. ideal framework, we show that all adversarial behavior in the real world (where there is no trusted party) can be simulated in the ideal world. In the following, we sketch the construction of such a *simulator* $\mathcal{S}$, while we treat the security of each building block (i.e., subroutine) of our construction separately and the overall security follows by the Composition Theorem for the semi-honest model [Gol04, Theorem 7.3.3].

Observe that in all our algorithms there are only two building blocks that act in the secret shared domain, the SumPub(.) and the InnerPub(.) algorithms. The rest of the computations are performed in the clear. The SumPub(.) algorithm is executed in one communication round. All the messages exchanged in this round are public inputs of the parties (independent of their private inputs). Thus, the views of the adversary and the simulator are exactly the same (i.e., indistinguishable) meaning that SumPub(.) is secure.

The InnerPub(.) algorithm starts with n invocations of the PRZS(.) function, where n is the number of items in each vector. This function was proposed and proven secure in [CDI05]. Next, (in the ideal world) the trusted party computes a share of the inner product per party, and adds to this, his share of a secret shared 0 value ($[0]$), which he obtained from the invocation of the PRZS(.) function. The adversary (in the real world) proceeds similarly, but computes the resulting share for each honest party on random shares (more precisely on shares of uniformly randomly integers), because it has no access to their private inputs. The addition of the secret shared 0 value at this step, restores the uniformly randomness of the shares. Thus, the "fresh" shares do not depend on the private inputs anymore. The aforementioned shares form the public inputs of the parties, which are accessible by both the adversary and the simulator. The indistinguishability of the adversary's (real-world) view from the simulator's (ideal-world) view follows from the fact that the adversary and simulator have identical views of the public inputs and indistinguishable views of the private inputs. The latter holds, because Shamir's shares are perfectly indistinguishable from random integers. Hence, the InnerPub(.) algorithm is proven secure. Having proven secure the SumPub(.) and the InnerPub(.) algorithms, by the composition theorem, the overall security of our protocols is guaranteed.

5.2 Performance Analysis

Our performance analysis presents the execution times of our verification algorithms. The implementation of these algorithms is based on VIFF [VD]. The experiments for timing our verification algorithms were conducted on an Intel(R) Core(TM) i3-2350M processor, at 2.3 GHz, with 4.00 GB RAM and Windows 7 64-bit operating system. We have conducted all tests on localhost, with 3 verifiers, and the network latency has not been taken into account. In VIFF, the prime p, determining the size of the field $\mathbb{Z}_p$, is selected to be

greater than $(2^{l+1} + 2^{l+k+1})$, where l is the maximum bit length of the inputs, and k is the statistical security parameter. These values are by default set to $l = 32$ and $k = 30$. For reasons of uniformity of the execution time results we use these default values for all our experiments. To handle fixed point numbers occurring in our setting, we scale them up to a precision of five decimal digits and then treat them as integers. The selection of $l = 32$ bits is large enough to handle this scaling both for our inputs and for our outputs.

In addition to the computational cost, determined by the number of variables, the number of entries per variable and the size p (of ~ 64 bits) of the field $\mathbb{Z}_p$, the communication cost also plays an important role in the overall performance. Recall that in our algorithms, the communication cost is brought only by the invocations of SumPub(.) and InnerPub(.), which are the only building blocks acting in the secret shared domain. The communication cost of each such invocation remains constant in all our algorithms, as for both subroutines only one aggregated value is sent, the size of which is upper bounded by the size of the field $\mathbb{Z}_p$, which is also constant and equal to 8 bytes (corresponding to the 64 bits of the prime p) per invocation, per verifier. Hence, our communication cost solely depends on the number of interactive rounds (i.e., the number of SumPub(.) and InnerPub(.) invocations). In the following, we deal with the round complexity of each statistic separately.

Mean and Variance: For the verification of the mean and variance we calculated those two statistics on the ages of 84 patients. One patient (out of the originally 85 patients) was excluded from the specific tests, because his age value was missing. The variable that influences the runtime is the number of patients $N = 84$ and the number of communication rounds is 1 and 2, for the mean and variance, respectively. The performance results of the Mean and Variance verification algorithms are presented in Table 1. These two tests do not scale perfectly linearly, due to their very small execution times.

	Mean	**Variance**
84 patients	43.5 ms	43.7 ms
168 patients	45.1 ms	49.7 ms
252 patients	45.8 ms	49.9 ms

Table 1: Performance of Mean and Variance Verification

ANOVA, Simple Linear Regression, Student's and Welch's t-tests: We conducted Welch's t-test on the time elapsed between receiving a message and reading it, versus patient compliance to that message. We have split our dataset of 2370 messages based on whether the patient complied to the message ($N_1 = 1404$) or not ($N_2 = 966$). Welch's t-test in this case, identifies whether there is a significant relationship between the compliance to a feedback message and the time elapsed between receiving it and reading it. It completes its execution in 4 rounds. For ANOVA we used the same time variable, versus the diagnosis (categorical variable taking 4 distinct diagnosis values) for the patient reading the feedback message in question. The dataset of 2370 messages is split based on the 4 different diagnoses to $N_1 = 953, N_2 = 524, N_3 = 365, N_4 = 528$ and requires 8 communication rounds. For simple linear regression, we examine the significance of the relationship between the aforementioned time variable (dependent variable), and the age of each patient (independent variable). We have excluded from our dataset the messages

concerning a patient for whom the age value was missing, and the size of our sample is $N = 2276$, while the number of required communication rounds is 4.

Our dataset does not contain two equal sized groups on which we can perform Student's t-test. Thus, we have excluded these runtimes from our performance results. Given the similarity of these two tests, their corresponding runtimes for the verification algorithms are also expected to be similar. Our performance results for Welch's t-test and F-test are summarized in Table 2; for regression they are given in Table 3. These tests scale linearly in the number of inputs (see Tables 2 and 3), as we have shown by doubling and tripling our dataset, and executing the algorithms on the augmented datasets.

	Welch's t-test	**F-test**
2370 msgs	165.5 ms	171.6 ms
4740 msgs	291.9 ms	315.1 ms
7110 msgs	404.1 ms	479.0 ms

Table 2: Performance of Welch's t-test and F-test Verification

	Regression
2276 msgs	304.3 ms
4552 msgs	586.4 ms
6828 msgs	884.6 ms

Table 3: Performance of Simple Linear Regression Verification

Chi-Squared test, Fisher's exact test and McNemar's test: We performed χ^2-test on the compliance to each of the 2370 feedback messages versus the diagnosis, to determine whether the data follows the χ^2 distribution. For Fisher's exact test, we split our dataset based on the gender of the 85 patients versus the diagnosis. We also performed and verified McNemar's test on the compliance to each of the 2370 feedback messages versus the feedback type (i.e., encouraging or discouraging message). Both the χ^2-test and Fisher's exact test verification require 8 communication rounds each, while McNemar's test requires 4 rounds. The performance results for χ^2-test and McNemar's test verification algorithms are given in Table 4, while the corresponding results for Fisher's test are presented in Table 5. As expected from our experimental setup, the execution time of the verification algorithms scales linearly in the number of input data.

	Chi-Squared	**McNemar's**
2370 msgs	207.3 ms	138.8 ms
4740 msgs	397.7 ms	238.8 ms
7110 msgs	594.4 ms	352.7 ms

Table 4: Performance of Chi-Squared and McNemar's test Verification

	Fisher's test
85 patients	53.3 ms
170 patients	57.7 ms
255 patients	70.3 ms

Table 5: Performance of Fisher's exact test Verification

6 Conclusion

We deal with privacy-preserving verification of statistics in clinical research, where the prover does not wish to disclose information about the inputs to the verifier. This is reciprocal to scenarios that previous works address, where the verifier does not wish to disclose

information about the inputs to the prover. We demonstrate that in the clinical research scenario under consideration, privacy-preserving verification of statistics can be performed so efficiently, that it can be applied in practice.

Acknowledgements: This work has been done in the context of the THeCS project which is supported by the Dutch national program COMMIT. We would like to thank Lorena Montoya and Job van der Palen for their help in statistics and clinical research.

References

[ABC+12] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, A. Shelat, and B. Waters. Computing on Authenticated Data. In *TCC*, pages 1–20. Springer, 2012.

[AIK10] B. Applebaum, Y. Ishai, and E. Kushilevitz. From Secrecy to Soundness: Efficient Verification via Secure Computation. In *ICALP*, pages 152–163. Springer, 2010.

[AodJH10] H. Akker op den, V. Jones, and H. Hermens. Predicting Feedback Compliance in a Teletreatment Application. In *ISABEL*, pages 1–5. IEEE, 2010.

[BCD+09] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure Multiparty Computation Goes Live. In *FC*, pages 325–343. Springer, 2009.

[BF11a] D. Boneh and D. Freeman. Homomorphic Signatures for Polynomial Functions. In *EUROCRYPT*, pages 149–168. Springer, 2011.

[BF11b] D. Boneh and D. Freeman. Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures. In *PKC*, pages 1–16. Springer, 2011.

[BGV11] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable Delegation of Computation over Large Datasets. In *CRYPTO*, pages 111–131. Springer, 2011.

[BLW08] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *ESORICS*, pages 192–206. Springer, 2008.

[BSMD10] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. In *USENIX*, pages 223–240. USENIX Assoc., 2010.

[CDI05] R. Cramer, I. Damgård, and Y. Ishai. Share Conversion, Pseudorandom Secret-Sharing and Applications to Secure Computation. In *TCC*, pages 342–362. Springer, 2005.

[CHd10] O. Catrina and S. Hoogh de. Secure Multiparty Linear Programming Using Fixed-Point Arithmetic. In *ESORICS*, pages 134–150. Springer, 2010.

[DA01] W. Du and M. J. Atallah. Privacy-Preserving Cooperative Statistical Analysis. In *ACSAC*, pages 102–110. IEEE, 2001.

[DF97] G. T. Duncan and S. E. Fienberg. Obtaining Information while Preserving Privacy: A Markov Perturbation Method for Tabular Data. In *JSM*, pages 351–362. IOS Press, 1997.

[DHC04] W. Du, Y. S. Han, and S. Chen. Privacy-Preserving Multivariate Statistical Analysis: Linear Regression and Classification. In *SIAM SDM*. Lake Buena Vista, Florida, 2004.

[DN04] C. Dwork and K. Nissim. Privacy-Preserving Datamining on Vertically Partitioned Databases. In *CRYPTO*, pages 134–138. Springer, 2004.

[Ens] M. Enserink. Stapel Affair Points to Bigger Problems in Social Psychology; http://news.sciencemag.org/scienceinsider/2012/11/final-report-stapel-affair-point.html.

[Fan09] D. Fanelli. How Many Scientists Fabricate and Falsify Research? A Systematic Review and Meta-Analysis of Survey Data. *PLOS ONE*, 4(5):e5738, 2009.

[Fie09] A. Field. *Discovering Statistics Using SPSS*. Sage Publications Limited, 2009.

[Fis22] R. A. Fisher. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *J. R. Stat. Soc.*, pages 87–94, 1922.

[GGP10] R. Gennaro, G. Gentry, and B. Parno. Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *CRYPTO*, pages 465–482. Springer, 2010.

[Gol04] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, 2004.

[GRR98] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. In *PODC*, pages 101–111. ACM, 1998.

[HR10] M. Hardt and G. N. Rothblum. A Multiplicative Weights Mechanism for Privacy-Preserving Data Analysis. In *FOCS*, pages 61–70. IEEE, 2010.

[JMSW02] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic Signature Schemes. In *CT-RSA*, pages 204–245. Springer, 2002.

[KLSR09] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter. Privacy-Preserving Analysis of Vertically Partitioned Data Using Secure Matrix Products. *JOS*, 25(1):125, 2009.

[McN47] Q. McNemar. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157, 1947.

[Md09] J. E. Muth de. Overview of Biostatistics Used in Clinical Research. *AJHP*, 66(1):70–81, 2009.

[Mis] Misconduct in Science: An Array of Errors; http://www.economist.com/node/21528593.

[OS08] B. R. Overholser and K. M. Sowinski. Biostatistics Primer: Part 2. *NCP*, 23(1):76–84, 2008.

[Pea00] K. Pearson. X. On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables Is Such that It Can Be Reasonably Supposed to Have Arisen from Random Sampling. *Lond. Edinb. Dubl. Phil. Mag.*, 50(302):157–175, 1900.

[PRV12] B. Parno, M. Raykova, and V. Vaikuntanathan. How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption. In *TCC*, pages 422–439. Springer, 2012.

[PST13] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of Correct Computation. In *TCC*, pages 222–242. Springer, 2013.

[RBG+00] J. Ranstam, M. Buyse, S. L. George, S. Evans, N. L. Geller, B. Scherrer, E. Lesaffre, G. Murray, L. Edler, J. L. Hutton, T. Colton, and P. Lachenbruch. Fraud in Medical Research: An International Survey of Biostatisticians. *Control clin trials*, 21(5):415–427, 2000.

[SCD+08] R. Sparks, C. Carter, J. B. Donnelly, C. M. O'Keefe, J. Duncan, T. Keighley, and D. McAullay. Remote Access Methods for Exploratory Data Analysis and Statistical Modelling: Privacy-Preserving Analytics. *Comput Meth Prog Bio*, 91(3):208–222, 2008.

[Sha79] A. Shamir. How to Share a Secret. *Comm. of the ACM*, 22(11):59–98, 1979.

[Swe02] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *Uncertain Fuzz*, 10(5):557–570, 2002.

[THH+09] B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao. Privacy-Preserving Computation and Verification of Aggregate Queries on Outsourced Databases. In *PETS*, pages 185–201. Springer, 2009.

[Tho] THORAX - An International Journal Of Respiratory Medicine; http://thorax.bmj.com/.

[VD] Team VIFF Developement. The Virtual Ideal Functionality Framework; http://viff.dk/.

[ZBT07] K. Zellner, C. J. Boerst, and W. Tabb. Statistics Used in Current Nursing Research. *Nurs Educ*, 46(2):55–59, 2007.

A Verification Algorithms

Algorithm 1 $v \leftarrow \mathsf{VrfMean}([\mathbf{x}], N, \bar{x})$

1: **Input:** [**x**] = secret shared vector **x**, N = size of sample group, $\bar{x}$ = calculated mean value to be verified
2: **Output:** v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
3: **for all** P_j **do**
4: **if** $(N \cdot \bar{x}) \neq \mathsf{SumPub}([\mathbf{x}])$ **then**
5: **return** 0
6: **return** 1

Algorithm 2 $v \leftarrow \mathsf{VrfVariance}([\mathbf{x}], \bar{x}, N, S^2)$

Input: $[\mathbf{x}]$ = secret shared vector $\mathbf{x}$, $\bar{x}$ = mean value of sample group x, N = size of sample group, S^2 = calculated variance to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if $\mathsf{VrfMean}([\mathbf{x}], N, \bar{x}) == 0$ **then**
  return 0
for all P_j **do**
  if $((N-1) \cdot S^2) \neq \mathsf{InnerPub}(([\mathbf{x}] - \bar{x}), ([\mathbf{x}] - \bar{x}))$ **then**
    return 0
return 1

Algorithm 3 $v \leftarrow \mathsf{VrfS\text{-}T\text{-}test}([\mathbf{x}], [\mathbf{y}], \bar{x}, \bar{y}, N, S_x^2, S_y^2, t)$

Input: $[\mathbf{x}]$ = secret shared vector $\mathbf{x}$, $[\mathbf{y}]$ = secret shared vector $\mathbf{y}$, $\bar{x}$ = mean value of sample group x, $\bar{y}$ = mean value of sample group y, N = size of sample group, S_x^2 = variance of sample group x, S_y^2 = variance of sample group y, t = t-value to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if $\mathsf{VrfVariance}([\mathbf{x}], \bar{x}, N, S_x^2) == 0$ **or** $\mathsf{VrfVariance}([\mathbf{y}], \bar{y}, N, S_y^2) == 0$ **then**
  return 0
for all P_j **do**
  if $(\bar{x} - \bar{y}) / (\sqrt{\frac{S_x^2 + S_y^2}{N}}) \neq t$ **then**
    return 0
return 1

Algorithm 4 $v \leftarrow \mathsf{VrfW\text{-}T\text{-}test}([\mathbf{x}], [\mathbf{y}], \bar{x}, \bar{y}, N_x, N_y, S_x^2, S_y^2, t)$

Input: $[\mathbf{x}]$ = secret shared vector $\mathbf{x}$, $[\mathbf{y}]$ = secret shared vector $\mathbf{y}$, $\bar{x}$ = mean value of sample group x, $\bar{y}$ = mean value of sample group y, N_x = size of sample group x, N_y = size of sample group y, S_x^2 = variance of sample group x, S_y^2 = variance of sample group y, t = t-value to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if $\mathsf{VrfVariance}([\mathbf{x}], \bar{x}, N_x, S_x^2) == 0$ **or** $\mathsf{VrfVariance}([\mathbf{y}], \bar{y}, N_y, S_y^2) == 0$ **then**
  return 0
for all P_j **do**
  if $(\bar{x} - \bar{y}) / (\sqrt{\frac{S_x^2}{N_x} + \frac{S_y^2}{N_y}}) \neq t$ **then**
    return 0
return 1

Algorithm 5 $v \leftarrow$ VrfF-test$(K, [\mathbf{x}][K], \bar{x}[K], N[K], S^2[K], F)$

Input: K = number of groups, [**x**] = table of K secret shared vectors $\mathbf{x}[K]$, $\bar{x}[K]$ = mean values of K sample groups x, $N[K] = K$ sizes of the sample groups, $S^2[K]$ = variances of the K sample groups, $F = F$-value to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
for $k = 1, ..., K$ **do**
  if VrfVariance$([\mathbf{x}]_k, \bar{x}_k, N_k, S_k^2) == 0$ **then**
    return 0
for all P_j **do**
  if $(\sum_{k=1}^{K} \frac{N_k(\bar{x}_k - \frac{\sum_{k=1}^{K} \bar{x}_k}{K})^2}{K-1}) / (\sum_{k=1}^{K} \frac{([\mathbf{x}]_k - \bar{x}_k)^2}{\sum_{k=1}^{K} N_k - K}) \neq F$ **then**
    return 0
return 1

Algorithm 6 $v \leftarrow$ VrfSimpleLinearRegression$([\mathbf{x}], [\mathbf{y}], \bar{x}, \bar{y}, \alpha, \beta, N)$

Input: [**x**] = secret shared vector **x**, [**y**] = secret shared vector **y**, $\bar{x}$ = mean value of sample group x, $\bar{y}$ = mean value of sample group y, $\alpha = \alpha$-value to be verified, $\beta = \beta$-value to be verified, N = size of sample group
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if VrfMean$([\mathbf{x}], N, \bar{x}) == 0$ **or** VrfMean$([\mathbf{y}], N, \bar{y}) == 0$ **then**
  return 0
for all P_j **do**
  if $\frac{\text{InnerPub}((([\mathbf{x}]-\bar{x}),([\mathbf{y}]-\bar{y}))}{\text{InnerPub}((([\mathbf{x}]-\bar{x}),([\mathbf{x}]-\bar{x}))} \neq \beta$ **or** $(\bar{y} - \beta \cdot \bar{x}) \neq \alpha$ **then**
    return 0
return 1

Algorithm 7 $v \leftarrow$ VrfContingency$([\mathbf{x}[C1]], [\mathbf{y}[C2]], C1, C2, F[C1, C2])$

Input: [**x**] = $C1 \cdot N$ secret shared bits of the unary representation of vector **x**, [**y**] = $C1 \cdot N$ secret shared bits of the unary representation of vector **y**, $C1$ = number of possible values under "cause" of the experiment, $C2$ = number of possible values under "effect" of the experiment, $F[C1, C2]$ = table of size $C1 \cdot C2$ containing the frequencies observed (i.e., inner part of the contingency table)
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
for $k = 1, ..., C1$ **do**
  for $l = 1, ..., C2$ **do**
    $VF[k, l] \leftarrow$ InnerPub$([\mathbf{x}_k], [\mathbf{y}_l])$
    for all P_j **do**
      if $VF[k, l] \neq F[k, l]$ **then**
        return 0
return 1

Algorithm 8 $v \leftarrow$ VrfChi-squaredTest$([\mathbf{x}[C1]], [\mathbf{y}[C2]], C1, C2, F[C1, C2], \chi^2)$

Input: [**x**] (resp. [**y**]) = bitwise secret shared vector **x** (resp. **y**), where each entry has $C1$ (resp. $C2$) bits, $C1$ = number of possible values under "cause" of the experiment, $C2$ = number of possible values under "effect" of the experiment, $F[C1, C2]$ = table of size $C1 \cdot C2$ containing the frequencies observed, $\chi^2 = \chi^2$-value to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if VrfContingency$([\mathbf{x}[C1]], [\mathbf{y}[C2]], C1, C2, F[C1, C2]) == 0$ **then**
  return 0
for all P_j **do**
  $model_{k,l} \leftarrow (\sum_{l=1,k=1}^{C2,C1} F_{k,l} \cdot \sum_{k=1,l=1}^{C1,C2} F_{k,l})/(\sum_{k=1}^{C1} RowTotal_k)$
  if $\sum_{k=1,l=1}^{C1,C2} \frac{(F_{k,l} - model_{k,l})^2}{modelk,l} \neq \chi^2$ **then**
    return 0
return 1

Algorithm 9 $v \leftarrow$ VrfFisher'sTest$([\mathbf{x}[C1]], [\mathbf{y}[C2]], C1, C2, F[C1, C2], p)$

Input: [**x**] (resp. [**y**]) = bitwise secret shared vector **x** (resp. **y**), where each entry has $C1$ (resp. $C2$) bits, $C1$ = number of possible values under "cause" of the experiment, $C2$ = number of possible values under "effect" of the experiment, $F[C1, C2]$ = table of size $C1 \cdot C2$ containing the frequencies observed, $p = p$-value to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if VrfContingency$([\mathbf{x}[C1]], [\mathbf{y}[C2]], C1, C2, F[C1, C2]) == 0$ **then**
  return 0
for all P_j **do**
  $VpUpper \leftarrow \prod_{k=1}^{C1} \sum_{l=1,k=1}^{C2,C1} F_{k,l}! \cdot \prod_{l=1}^{C2} \sum_{k=1,l=1}^{C1,C2} F_{k,l}!$ #numerator of the p fraction
  $VpLower \leftarrow \prod_{k=1,l=1}^{C1,C2} F_{k,l}! \cdot \sum_{k=1}^{C1} \sum_{l=1,k=1}^{C2,C1} F_{k,l}!$ #denominator of the p fraction
  if $\frac{VpUpper}{VpLower} \neq p$ **then**
    return 0
return 1

Algorithm 10 $v \leftarrow$ VrfMcNemar'sTest$([\mathbf{x}[2]], [\mathbf{y}[2]], F[2, 2], \chi^2)$

Input: [**x**] (resp. [**y**]) = bitwise secret shared vector **x** (resp. **y**), where each entry has 2 bits, $F[2, 2]$ = 2×2 table of frequencies, $\chi^2 = \chi^2$-value to be verified
Output: v = 0 or 1; 0 $\rightarrow$ unsuccessful verification, 1 $\rightarrow$ successful verification
if VrfContingency$([\mathbf{x}[2]], [\mathbf{y}[2]], 2, 2, F[2, 2]) == 0$ **then**
  return 0
for all P_j **do**
  if $\frac{(F_{1,2} - F_{2,1})^2}{F_{1,2} + F_{2,1}} \neq \chi^2$ **then**
    return 0
return 1