Custodes: Auditable Hypothesis Testing

Sacha Servan-Schreiber
Brown University
Providence, RI, USA
aservans@cs.brown.edu

Olga Ohrimenko
Microsoft Research
Cambridge, UK
oohrim@microsoft.com

Tim Kraska
MIT CSAIL
Cambridge, MA, USA
kraska@mit.edu

Emanuel Zgraggen
MIT CSAIL
Cambridge, MA, USA
emzg@mit.edu

ABSTRACT

"Quis custodiet ipsos custodes?"* – Juvenal, *Satires VI.

We present Custodes: a new approach to solving the complex issue of preventing "p-hacking" in scientific studies. The novel protocol provides a concrete and publicly auditable method for controlling false-discoveries and eliminates any potential for data dredging on the part of researchers during data-analysis phase. Custodes provides provable guarantees on the validity of each hypotheses test performed on a dataset by using cryptographic techniques to certify outcomes of statistical tests. Custodes achieves this using a decentralized authority and a tamper-proof ledger which enables the auditing of the hypothesis testing process. We present a construction of Custodes which we implement and evaluate using both real and synthetic datasets on common statistical tests, demonstrating the effectiveness and practicality of Custodes in the real world.

1 INTRODUCTION

"Data is the new oil" and as such, it is mined (i.e., gathered), shared, analyzed, re-analyzed, and re-re-analyzed until it yields to more and more interesting insights. With every exploration to find yet another insight, the chance of encountering a random correlation increases. This phenomenon is formally known as the multiple comparisons problem (MCP) and, if done in a systematic fashion, is often referred to as "HARKing" [32], "p-hacking" [23] or "data dredging".

While a variety of statistical techniques exist to control the false discovery rate (FDR) [3, 15], there is surprisingly almost no support to ensure that analysts actually use them. Rather individual research groups rely on often varying data analysis guidelines and trust in their group members to follow them. Things get even worse when the same data is analyzed by several institutions or teams. It is currently close to impossible to reliably employ statistical procedures, such as the Bonferroni [15] method, that guard against p-hacking across collaborators. It only requires one member to "misuse" the data (whether intentionally or not) and detecting, let alone recovering from such incidents is next to impossible. This problem is perhaps amplified by three factors: 1) the pressure on PhD students and PIs to publish [37], 2) "publication bias" [13] as papers with significant results are more likely to be published, and 3) the increasing trend to share and make datasets publicly available for any researchers to use. It is therefore unsurprising that the MCP is among the leading reasons why the scientific community is plagued by false discoveries [2, 25–27].

To illustrate this problem further consider a publicly available dataset such as MIMIC III [29]. This dataset contains de-identified health data associated with $\approx$ 40,000 critical care patients. MIMIC III has already been used in various studies [20, 24, 35] and it is probably one of the most (over)analyzed clinical datasets. As such, any discovery made on MIMIC runs the risk of being a false discovery. Even if a particular group of researchers follow a proper FDR protocol, there is no control over happens across different groups and tracking hypotheses at a global scale poses many challenges. It is therefore hard to judge the validity of any insight derived from such a dataset.

A solution to guarantee validity of insights commonly used in clinical trials - preregistration of hypotheses [10] - falls short in these scenarios. The data is collected upfront without knowing what kind of analysis will be done later on. Perhaps more promising is the use of a hold-out dataset. The MIMIC authors can release only 30K patient records as an exploration dataset (EDS) and hold back 10K as a validation dataset (VDS). The EDS can then be used in arbitrary ways to find interesting hypotheses. However, before any publication is made by a research group using the dataset, all hypotheses must be tested for its statistical significance over the VDS. Unfortunately, in order to use the VDS more than once, the same requirement as before holds true: every hypothesis over the VDS has to be tracked and controlled for. Furthermore, the data owner, the MIMIC authors in this case, need to provide this hypothesis validation service. This is both a burden for the data owners as well as a potential

risk. Researchers need to trust the data owners to apply FDR control procedures correctly and to objectively evaluate their hypotheses.

The above example illustrates the motivation behind Custodes. Our goal is to create a system that guarantees the validity statistical test outcomes and allows readers (and/or reviewers) of publications to audit them for correctness. Using proven cryptographic techniques to certify outcomes of statistical tests by a decentralized authority, we eliminate the risk of data-dredging (intentional and otherwise) on the part of researchers or data owners. Custodes can be used in various settings, including cases where the data is public and only the hold-out data is fed into Custodes (as in the example above), in smaller settings where a few research groups collaborate on combined data, or even within single teams where lab managers can opt to encrypt all of their data and use Custodes as a way to prevent unintentional false discoveries, assign accountability and foster reproducibility.

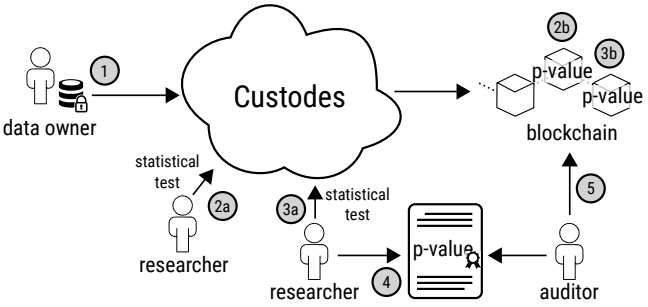

Figure 1: High-level overview of Custodes.

2 CONTRIBUTIONS

The primary contribution of this work is the Custodes framework. Figure 1 shows a high-level overview of the proposed system. (1) A data owner encrypts their dataset (either the full dataset or just a hold-out) and submits it to Custodes: a decentralized platform consisting of multiple nodes that are run by different entities (e.g., universities or research groups). (2a and 3a) Researchers can post requests for statistical tests to Custodes. (2b and 3b) Custodes securely computes these tests using several cryptographic techniques, and stores the results (and transcript of the computation) sequentially on a tamper-proof ledger (e.g., a Blockchain [36]). (4) One of the researchers decides to publish their finding and includes a "certified p-value" in their paper. (5) A reader or auditor of this publication can query the ledger, retrieve all p-values of all the tests that have been run on this particular dataset and apply an incremental FDR control procedure *a posteriori* [18, 45] to validated the publication's finding.

Custodes prevents p-hacking by using encrypted data and by guaranteeing that every statistical test is accounted for. It is, to the best of our knowledge, the first cryptographically based solution to the problem of p-hacking with provable security guarantees. Custodes is a framework which we instantiate based on additively homomorphic encryption and multi-party computation protocols. Using those building blocks, we implement three widely used hypothesis tests (Student's T-test, Pearson Correlation, and Chi-Squared) in Custodes and evaluate its performance with various configurations and dataset sizes.

3 SYSTEM OVERVIEW

We model the scenario considered in the introduction as follows: a data owner wants to release a dataset $\mathcal{D}$, to a set of researchers, denoted $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, for the purpose of executing statistical tests. Note: $\mathcal{D}$ can be either a full dataset or a hold-out. We aim to build a system that guarantees that all statistical tests over $\mathcal{D}$ are accounted for and are executed and reported in a truthful manner. In other words, an MCP control procedure is correctly applied for *all* statistical tests computed on $\mathcal{D}$. If all p-values resulting from tests computed on $\mathcal{D}$ are accounted for, then it is possible to apply an MCP control procedure to determine which null-hypothesis should be accepted (resp. rejected) to ensure the probability of a false-rejection remains below a threshold [3, 18, 45] (detailed in Section 3). Moreover, the entire process must be auditable: a publication can be vetted for having followed the correct MCP-control procedure. At a high level, Custodes achieves these requirements by restricting the exposure of the dataset $\mathcal{D}$ such that researchers are only able to test hypotheses on $\mathcal{D}$ via the Custodes interface which stores all hypotheses and results in a tamper-proof, auditable trace.

3.1 Custodes Design

Our design of Custodes is motivated by the following observations. The data owner cannot release a dataset $\mathcal{D}$ to the researchers directly since it creates a possibility for p-hacking and other forms of bias (e.g., parties can run tests privately and report only favorable results without controlling for the MCP). Hence, $\mathcal{D}$ has to be released such that only the output of the statistical test performed on $\mathcal{D}$ is made available to researchers. A naïve solution is to provide access to $\mathcal{D}$ through a stringent interface which *only* returns the results of statistical tests. However, even such a solution is not sufficient for enforcing correctness as it can be abused by parties querying the interface until a favorable (i.e., significant) result is obtained while avoiding to report the intermediate queries or adequately controlling for the MCP. Moreover, such a solution has no way to audit the test computation making it impossible to "certify" the discoveries.

Custodes addresses this problem by using several cryptographic methods, specifically, *secure computation* and a

tamper-proof ledger both of which we formally introduce in the following section. At a high level, Custodes requires that a dataset $\mathcal{D}$ be encrypted by the data owner *prior* to being made available to researchers (or publicly released). Researchers use the encrypted dataset, denoted $[\mathcal{D}]$, to compute statistical tests using secure computations over the encrypted data with a mechanism for revealing (decrypting) only the test statistic while simultaneously ensuring the MCP is controlled for. Custodes ensures that each revealed statistic is recorded on a publicly accessible and secure ledger making each result of a statistical test public and verifiable. This latter requirement ensures that 1) all statistical tests executed on $[\mathcal{D}]$ are recorded *in sequence* and 2) makes it possible to certify that MCP control procedures are applied correctly. We elaborate on these two important requirements in subsequent sections.

3.2 Security Goals

At a high level, Custodes aims to achieve the following security goals. We formalize these properties in Section 6.

- **Correctness.** Statistical tests computed on $[\mathcal{D}]$ are correct, i.e, the p-value of a test computed in Custodes is equivalent to the p-value computed through standard statistical packages.
- **Confidentiality.** The only information revealed about $\mathcal{D}$ is the results of statistical tests executed through Custodes which ensures that the MCP is fully controlled for and no hypothesis can be tested outside of Custodes's interface from "leaked" information about $\mathcal{D}$.
- **Access control.** Statistical tests on $\mathcal{D}$ can be executed only by a set of approved researchers which ensures accountability in the hypothesis testing procedure.
- **Verifiability.** Every hypothesis test executed through Custodes has a corresponding certificate ψ that is publicly accessible and verifiable, even by a third-party.
- **Auditability.** Given a certificate ψ, anyone can verify the correctness of the hypothesis test computation associated with the certificate (i.e., determine whether the null hypothesis is true or false).

3.3 Threat Model

Custodes provides provable guarantees to the five properties outlined in Section 3.2 under the following threat model. Our security assumptions are with respect to the data owner, participants engaged in the protocol, and the network over which the system is instantiated.

Data Owner. We assume that the data owner does not collude with the researchers who run the statistical tests. This assumption is necessary given that a malicious owner has unfettered access to the (unencrypted) dataset $\mathcal{D}$ and can thereby trivially bypass any MPC control procedure.

Participants. We assume that all parties are *honest-but-curious*, that is, they individually adhere to the correct protocol but are interested in learning more about the underlying dataset so that they may circumvent MCP control procedures, and may collude among themselves in attempt to achieve this (i.e., to engage in "p-hacking"). To this end, we assume that at most $t-1$ out of n parties may collude with each other in order to obtain more information about the dataset or manipulate a test certificate. In addition, we assume that the set of colluding parties is static (i.e., does not change during protocol execution). Finally, we require a secure public key infrastructure in place which allows to identify individual parties (researchers) by their public key and assume that all messages exchanged during protocol execution are digitally signed against a party's identity.

Network. Custodes does not rely on a secure communication channel between participants in the network and therefore tolerate the presence of a *passive* adversary controlling the network. However, we do not assume the presence of an *active* network adversary (one that can actively block or otherwise alter communication) as that would disrupt the availability of participants and the tamper-proof ledger. We note that this requirement allows for messages exchanged between parties to be made publicly available (e.g., by recording them on a public ledger) without compromising security of the overall system.

3.4 Controlling False Discoveries

An important step in realizing the construction of Custodes is understanding how MCP control procedures are used in the data analysis phase.

In order to control for false discoveries it is necessary to know all p-values computed over dataset up to the current test. While standard procedures such as Bonferroni [15] must be applied *a posteriori* of the data analysis (once all hypotheses have been tested), in Custodes, we desire a method for controlling the FDR in a streaming fashion, ideally without knowledge or restriction on future tests. We use a common control procedure known as α-investing [18]. The α-investing procedure is a standard choice for controlling false discoveries in practice when the total number of hypotheses is unknown beforehand [45, 46]. We briefly describe the procedure below and we refer the reader to [18, 45] for additional details to the ones we provide as they are outside the scope of this work.

Formally, α-investing controls the *marginal False Discovery Rate* (mFDR) which is defined as:

$$mFDR_\eta(i) = \frac{E[V(i)]}{E[R(i)] + \eta}$$

where i denotes the total number of tests which have been executed, while $V(i)$ (resp., $R(i)$) denote the number of false (resp., total) discoveries after i hypotheses have been tested using the α-investing procedure. A testing procedure controls the $mFDR_\eta$ at level α if $mFDR_\eta(i) \leq \alpha$ where the parameter η (e.g., $\eta = (1-\alpha)$ [45]) weighs the impact in case of few discoveries. Under the complete null hypothesis $V(i) = R(i)$ and hence $mFDR_\eta(i) \leq \alpha$ implies that $E[V(i)] \leq \alpha\eta/(1-\alpha)$. If we chose $\eta = 1 - \alpha$ then $E[V(i)] \leq \alpha$ and it is possible to conclude that the false discovery rate is controlled at level α [18]. Intuitively the α-investing procedure works by assigning to each hypothesis test a budget α' from an initial "*α-wealth*." If the p-value of the null hypothesis being considered is above α' the null hypothesis is accepted and some budget is lost, otherwise it is rejected and some testing budget is gained. In Custodes, the *α-wealth*, α and η can be determined by the data owner or set as a static constant. The importance of this procedure to this work is that given all p-values computed on $\mathcal{D}$ up to the ith test, it is easy to determine whether the hypothesis of the $i+1$st test should be accepted (resp. rejected) based on the resulting p-value.

3.5 Certification of Hypotheses

The overarching goal of Custodes is to certify insights gained from data as valid. Therefore, Custodes must provide a mechanism for certifying the outcomes of hypotheses testing procedures. Taking a birds-eye view, for every test that Custodes executes, it produces a publicly available certificate ψ which is verifiable and unforgeable. Formally, we define a certificate of a statistical test as a tuple $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, (\mathsf{t}, \mathsf{p}))$ where τ is the *test index* (i.e., its order among all the tests that have been executed so far on $\mathcal{D}$), $\mathcal{P}^{\mathsf{pk}}$ is the identifier of the researcher that requested the test (e.g., a public key), $\mathcal{T}$ is the code of the statistical test function, and (t, p) is the result of executing $\mathcal{T}(\mathcal{D})$ where t is the test statistic and p is the p-value corresponding to t. Given that each certificate contains both the test index τ and resulting p-value, it is trivial to verify whether an MCP control procedure was applied when accepting (resp. rejecting) a null-hypothesis using the procedure in Section 3.4. Note that the certificate by itself does not ensure that $(\mathsf{t}, \mathsf{p}) = \mathcal{T}(\mathcal{D})$, however, Custodes records sufficient information on a tamper-proof ledger that anyone can verify whether it is indeed the case. We formalize and elaborate on these claims in Section 5.

4 CUSTODES PROTOCOL

Now that the design and security goals of Custodes have been made explicit, we are ready to describe the ideas of the framework at a high level (Section 4.1), afterwards, we introduce the cryptographic building-blocks (Section 4.2), and then describe the concrete instantiation of Custodes in Section 4.3. We note that, though the cryptographic techniques used as building-blocks are well known, they have to be combined in a new and careful manner to achieve a *auditible MCP-control protocol*. (See Table 1 for a summary of properties these building blocks help us to achieve.)

4.1 Framework

The functionality of Custodes framework can be split into three phases: *Setup*, *Compute*, and *Audit*.

Setup. A data owner, releases an (encrypted) dataset $[\mathcal{D}]$ and shares the secret key used to encrypt $\mathcal{D}$ with parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ such that any threshold number of them can collectively decrypt $[\mathcal{D}]$. The data owner then initializes a Blockchain $\mathcal{B}$ and posts to $\mathcal{B}$ a signed message $(0, \mathsf{pk}, \perp, \perp)$ that corresponds to the counter of the tests to be executed on $\mathcal{D}$ set to zero. In addition to releasing the encrypted dataset, the data owner releases metadata pertaining to $\mathcal{D}$ deemed sufficient for researchers to form hypotheses on the dataset. We formalize the metadata requirements in Section 6.

Compute. A researcher, say $\mathcal{P}$, specifies the test $\mathcal{T}$ corresponding to a statistical test that she wants to execute on $\mathcal{D}$. We note that $\mathcal{P}$ is one of the parties in $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ approved by the data owner during the setup of Custodes. $\mathcal{P}$ posts a message $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, \perp)$ to $\mathcal{B}$ and obtains the *test index*, τ. For simplicity, we assume that only one test is executed at a time though we note that this is not a necessary requirement since the dataset is static and $\mathcal{B}$ ensures that every test is assigned a sequential test index τ. Once the test is computed, the result of the test, (t, p), is made available only in an encrypted form as a result of *secure computation*. Recall that the test result cannot be recovered by *any* of the parties individually and requires a threshold number of the parties reach consensus in order to reveal it. To this end, $\mathcal{P}$ requests help from $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ by posting a signed message to $\mathcal{B}$ along with the (encrypted) result. Each party $\mathcal{P}_i \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ verifies that the message indeed came from one of the researchers allowed to run the tests and posts a message to $\mathcal{B}$ with a partially decrypted share of the (t, p). When combined, the shares reveal the decrypted result. This specification ensures that each test result is made publicly available (to all parties). The final entry recorded on $\mathcal{B}$ is the test certificate consisting of the tuple $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, (\mathsf{t}, \mathsf{p}))$. This final tuple is signed by all parties engaged in the reveal computation (i.e., the parties revealing the result) since after the shares are revealed,

Property	Cryptographic Building block(s)	Role in Custodes
Access Control	Digital signatures & threshold encryption.	Only approved researchers can request test computation and a threshold majority of approved parties can decrypt the data.
Hypothesis Tracking	Threshold encryption, multi-party computation, tamper-proof ledger.	Computation result can only be revealed through a consensus of parties in the system which record every tested hypothesis at decryption time on a ledger.
Auditability	Tamper-proof ledger	A transcript of every statistical test computation is recorded on the ledger making them verifiable and auditable by researchers and third-parties.
Certification	Tamper-proof ledger	Sequence of statistical tests recorded on the ledger n make it possible to apply the α-investing procedure *a posteiori* which serves as a certificate.

Table 1: Summary of cryptographic building blocks and their role in realizing the main properties of Custodes.

each party may individually reconstruct and ensure that the shares indeed correspond to the correct decryption.

Audit. A test with certificate $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, (\mathsf{t}, \mathsf{p}))$ can be audited for correctness by any entity with access to $\mathcal{B}$, $[\mathcal{D}]$ and the public key pk used to encrypt $\mathcal{D}$. An auditor $\mathcal{V}$ retrieves all messages related to τ from $\mathcal{B}$ and verifies the signature of each message, including the certificate. $\mathcal{V}$ then proceeds to use this information to ensure the test code was computed correctly and indeed produces the result (t, p). We require that $\mathcal{B}$ contains sufficient information to verify whether $(\mathsf{t}, \mathsf{p}) = \mathcal{T}(\mathcal{D})$, even if all parties are offline, while preventing any information be revealed about $\mathcal{D}$ (except for the result of the statistical test). In particular, any entity with access to $\mathcal{B}$ (e.g., a researcher, an auditor, a data owner, or a third party) can verify the computed statistical tests that have been run through Custodes: for every tuple $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, (\mathsf{t}, \mathsf{p}))$ anyone can verify that (t, p) is the correct output Hence, if $\mathcal{T}(\mathcal{D}) \neq (\mathsf{t}, \mathsf{p})$, $\mathcal{P}$'s malicious behaviour is exposed and the result (t, p) is deemed invalid.

4.2 Cryptographic Building-Blocks

Our Custodes instantiation uses three common building-blocks from the cryptographic literature: additively-homomorphic encryption, a tamper-proof ledger, and secure multi-party computation. (We note that Custodes is a framework and could be instantiated using, for example, fully homomorphic encryption to perform secure computation instead. However, for the statistical tests considered in this paper, additively homomorphic encryption was sufficient and remains more efficient in practice.)

Additively-Homomorphic Encryption. To encrypt a dataset in a way which enables performing computations over the data we use additively-homomorphic encryption. This is a special form of encryption which enables the evaluation of a subset of arithmetic operations over encrypted values without knowledge of the secret key used to encrypt. Such schemes make it possible to evaluate a subset of functions without revealing information on the inputs nor output. While there are several different schemes to choose from, we use the additively homomorphic scheme known as Paillier [38] as it easily supports threshold decryption and its functionality can be augmented using multi-party computation [11].

Augmenting Paillier functionality. Paillier enables the evaluation linear arithmetic gates (i.e., addition, subtraction) out-of-the-box but does not support non-linear arithmetic evaluations such as multiplication and division. However, it is possible to augment the functionality of Paillier to support non-linear arithmetic using secure multi-party computations which we describe next. We used methods described in [6, 7, 12] to compute non-linear arithmetic gates using interactive secure multi-party computation (MPC) protocols (e.g., multiplication and division gates). We emphasize that while the protocols used in this work were originally described for secret-sharing schemes (i.e., [39]), all protocols apply to a threshold-Paillier setting, even when active security is required [11]. We elaborate on the details of these protocols in Section 5.

Tamper-proof Ledger. We rely on a tamper-proof ledger abstraction which refer to as a Blockchain, $\mathcal{B}$. The Blockchain records statistical test executed on the dataset such that each test result, order of test execution, and the identity of the researcher requesting execution is publicly available and immutable. $\mathcal{B}$ can be maintained by a central authority (e.g., the

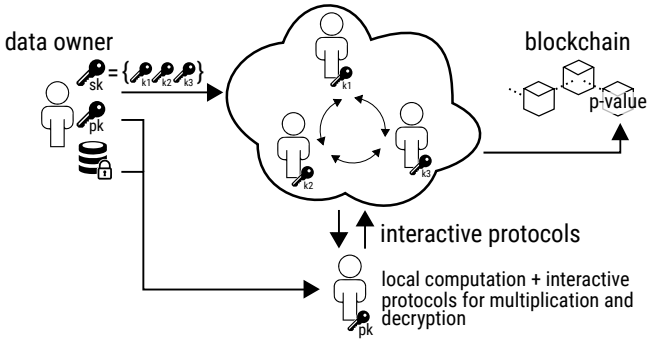

Figure 2: Overview of the threshold Paillier based Custodes instantiation (Section 4.3). The researcher receives an encrypted dataset from the data owner and proceeds to compute statistical tests using both local computations and interactive protocols, using parties in the Custodes network for computing non-linear functions and to certify the obtained p-value(s) on the blockchain.

data owner), by the parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ themselves using a consensus protocol (e.g., Paxos [33]), or by independent parties (e.g., using a public Blockchain based on a decentralized consensus protocol such as Nakamoto consensus [36]). Similar to the ledger abstraction used in [8], our main requirement is that the above abstraction is correct and always available[1].

4.3 Protocol Implementation

We are now ready to describe the detailed construction of Custodes. By using a threshold somewhat-homomorphic encryption scheme for encrypting the dataset, and distributing the key among a set of (possibly untrusted) parties, we can construct Custodes with minimal overhead on the researchers engaged in the system. Indeed, we achieve a construction which only requires the active participation of parties in the network in two cases: 1) when computing a non-linear arithmetic gate in the statistical test circuit evaluated over the encrypted dataset $[\mathcal{D}]$ (as explained above) and 2) for decryption of the result of the computation (i.e., to reveal the test statistic).

Recall that we divide the overall protocol into three distinct phases (Setup, Compute and Audit).

Setup. During the setup phase, the data owner executes algorithm $\texttt{Paillier.KeyGen}(1^k)$ that outputs a public key pk and secret key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$. The shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ are distributed to parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, respectively. The data owner then runs $\texttt{Encrypt}(\mathsf{pk}, \mathcal{D})$ on dataset $\mathcal{D}$ that returns encoded dataset $[\mathcal{D}]$, where every value of $\mathcal{D}$ is individually encrypted. The data owner sends the tuple $(\mathsf{pk}, \mathsf{sk}_i)$ to $\mathcal{P}_i \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, $\forall i = 0 \ldots n$ and makes $[\mathcal{D}]$ publicly available to all parties. The data owner then initializes a Blockchain $\mathcal{B}$ and records the digital identity of every party $\mathcal{P}_i \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ that can run the protocol (e.g., it can be the verification key of a digital signature that will be used to verify $\mathcal{P}_i$'s signatures) on $\mathcal{B}$. The owner then posts the signed message $(0, \mathsf{pk}, \perp, \perp)$ corresponding to the counter of the tests to be executed on $\mathcal{D}$ set to zero. In addition to releasing the dataset, the data owner makes available metadata pertaining to the dataset (e.g., size of the dataset, attributes, etc). With the exception of dataset size, the choice of metadata to be released is left up to the data owner.

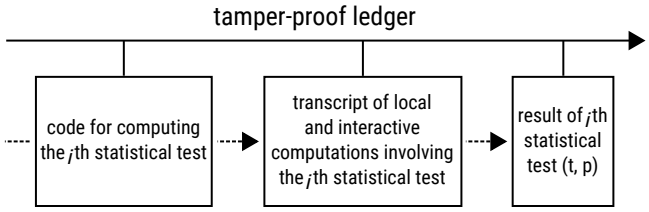

Figure 3: Use of Tamper-proof ledger in Custodes.

Compute. Let $\mathcal{P}$ be the researcher that wishes to run a statistical test $\mathcal{T}$ represented as an arithmetic circuit with input $[\mathcal{D}]$. $\mathcal{P}$ posts $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, \perp)$ to $\mathcal{B}$ and obtains counter τ. $\mathcal{P}$ then proceeds to deterministically evaluate the arithmetic test circuit $\mathcal{T}([\mathcal{D}])$ locally by using the homomorphic properties of the Paillier scheme. Assuming the arithmetic circuit contains a set of non-linear gates, $\mathcal{P}$ is only able to evaluate the first $w - 1 \geq 0$ gates locally before reaching some gate w which requires an interactive protocol. Let $\mathsf{op}_w(C_w)$ denote this computation where op_w is the arithmetic gate and C_w is the tuple of ciphertexts that $\mathcal{P}$ obtained from local computations up to level $w - 1$. Suppose, op_w is a multiplication gate and C_w contains ciphertexts $[a]$ and $[b]$. $\mathcal{P}$ posts to $\mathcal{B}$ the message tuple $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathsf{op}_w, C_w)$. All parties receive this message and proceed to engage in the interactive protocol $\texttt{Mult}$ to compute $\mathsf{op}_w(C_w)$ and obtain the encrypted result of this computation $[c] = [ab]$. Every party $\mathcal{P}_i$ then posts a signed message tuple $(\tau, \mathcal{P}_i^{\mathsf{pk}}, \mathsf{op}_w, [c])$ to $\mathcal{B}$. $\mathcal{P}$ then uses [c] and proceeds with the local computation(s). In general, for the jth interactive protocol required during the evaluation of $\mathcal{T}$, $\mathcal{P}$ posts a message $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathsf{op}_j, C_j)$ and the parties engage in the protocol, posting the result of op_j to $\mathcal{B}$. Finally, once the evaluation of $\mathcal{T}([\mathcal{D}])$ is completed, $\mathcal{P}$ requests the decryption of the result $([\mathrm{t}], [\mathrm{p}]) = \mathcal{T}([\mathcal{D}])$ by posting the tuple $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, ([\mathrm{t}], [\mathrm{p}]))$ to $\mathcal{B}$. Once the result it obtained, the certificate $\psi = (\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, (\mathrm{t}, \mathrm{p}))$ can be generated from the information recorded on $\mathcal{B}$ which, in conjunction with the

[1] In general, this is standard requirement for tamper-proof ledgers.

information posted to $\mathcal{B}$, certifies the result of the hypothesis test (t, p). Figure 3 illustrates the use of the tamper-proof ledger in the Compute process.

Audit. Suppose an auditor $\mathcal{V}$ wishes to verify that the certificate $\psi = (\mathcal{P}^{\mathsf{pk}}, \tau, \mathcal{T}, (\mathrm{t}, \mathrm{p}))$ indeed certifies the result (t, p). $\mathcal{V}$ retrieves $[\mathcal{D}]$ and proceeds to evaluate the test circuit $\mathcal{T}([\mathcal{D}])$ until the first non-linear operation (which required an interactive protocol during the Compute phase). Let op'_w denote the first such operation and C'_w denote its ciphertext list. $\mathcal{V}$ retrieves $(\tau, \mathcal{P}^{\mathsf{pk}}, \mathrm{op}_w, C_w)$ from $\mathcal{B}$ and verifies that $\mathrm{op}' = \mathrm{op}$ and $C'_w = C_w$. $\mathcal{V}$ then gathers the transcript resulting from the computation of op_w from $\mathcal{B}$ and reconstructs $[c]$. Note that $\mathcal{V}$ does *not* engage in an interactive protocol to obtain $[c]$. $\mathcal{V}$ then proceeds with the evaluation of $\mathcal{T}$ using $[c]$ as input and continues these verification steps for every gate in $\mathcal{T}$ until obtaining a result $([\mathrm{t}], [\mathrm{p}])' = ([\mathrm{t}], [\mathrm{p}])$ at which point $\mathcal{V}$ accepts the certificate ψ as valid. Conversely, if $([\mathrm{t}], [\mathrm{p}])' \neq ([\mathrm{t}], [\mathrm{p}])$, $\mathcal{V}$ rejects the certificate. We omit the verification of ciphertexts and other verification since we assume all parties adhear to protocol. However, we note that there exist numerous ways to augment the audit procedure by requiring various cryptographic proofs from the computing parties as described in [11].

Finally, once the certificates have been verified, $\mathcal{V}$ can verify whether the MCP control procedure described in Section 3.4 were applied correctly when accepting (resp. rejecting) the τth hypothesis by ensuring $mFDR_\eta(\tau) \leq \alpha$ for the specified η and α parameters.

5 STATISTICAL TESTS

The previous section showed how a dataset can be encrypted to ensure that every tested hypothesis is locked (i.e., by being encrypted) such that the MCP control procedure can be enforced in an auditable way. However, one important piece remains: How do researchers execute statistical tests over the encrypted dataset in an efficient way. In this section we describe the techniques from several lines of work surrounding secure computation which we combine in a secure way to realize protocols for computing statistical tests. We present pseudo code for computing the three common statistical tests: Student's T-test, Pearson Correlation and Chi-Squared. While not exhaustive, this trio of statistical tests forms a basis for quantitative analysis and covers many of use cases: from reasoning about population means to analyzing differences between sets of categorical data. We stress, however, that Custodes can be easily extended to other statistical tests using similar techniques.

The pseudo-code in this section computes the test statistics of Student's T-test, Pearson Correlation and Chi-Squared, t, r and χ^2, respectively. We note that using these statistics computing p-values can be done trivially for each respective statistic using the *degrees of freedom* [14, 31] associated with the data and we therefore omit the full details of this process. Interestingly, computing p-values does not require knowing the *sign* of the test statistic [14]. We exploit this fact to avoid computationally expensive secure comparisons but note that it is possible to extract the sign of each statistic should it be necessary for other purposes [6, 47].

5.1 Dataset Characteristics and Notation

Before diving into the implementation details of individual tests, we first describe how the data is structured in Custodes and introduces some notation.

Let $\mathbb{D}_{k\times w}(\mathbb{R})$ be the set of all real-valued k-by-m matrices. Formally, a dataset $\mathcal{D} \in \mathbb{D}_{k\times w}(\mathbb{R})$ is a matrix with k rows and m columns (attributes). Recall that the encrypted form of $\mathcal{D}$ is denoted by $[\mathcal{D}]$ where each entry is encrypted. Then $[\mathcal{D}] \in \mathbb{D}_{k\times w}(\mathbb{Z}_{N^2})^2 of this work$. Therefore, $\mathcal{D}$ can be seen as a $k\times w$ matrix containing real values and $[\mathcal{D}]$ as a $k\times w$ matrix containing encrypted fixed-point approximations to the real values of each entry. In describing the tests, we assume *wlog*, that the tests are computed over the first w (where $w \geq 2$) attributes $\pi_1, \ldots, \pi_w$ of $\mathcal{D}$. As such, we denote the value of the ith row of attribute j as $\mathcal{D}_{i,\pi_j}$, equivalently denoted as $[\mathcal{D}_{i,\pi_j}]$ in the encrypted dataset.

To allow researchers to form hypotheses on $\mathcal{D}$, we require the data owner to release *attribute metadata* (e.g., number of attributes, their type and domain size, independence from other attributes, etc) and assume that the size of the dataset is known. We define $\mathcal{L}_\mathrm{s} : \mathbb{D}_{k\times w} \to \{0,1\}^*$ to be a function from datasets to binary strings encoding attribute metadata. Given $\mathcal{L}_\mathrm{s}(\mathcal{D})$ it should be possible to 1) define a hypothesis on $\mathcal{D}$ and 2) perform a statistical test in Custodes. At minimum, we require that $\mathcal{L}_\mathrm{s}(\mathcal{D})$ provides k, m, a set of independent attributes and a set of bounds on attribute domains.

5.2 Interactive Protocols

After forming a hypotheses, researchers must be able to evaluate the statistic test over the encrypted dataset. Recall that the dataset is encrypted using the Paillier encryption scheme which allows them to evaluate a portion of the statistical test locally. However, to compute the full test circuit, it is necessary to engage in interactive protocols (see Section 4.2). Concretely, there are four interactive protocols needed to evaluate arbitrary statistical tests using the Paillier scheme. We list and describe each protocol below but omit the full details of their implementation as each one is extensively described in [6, 7, 12].

[2]The ciphertext space in the Paillier scheme is the ring of integers $\mathbb{Z}_{N^2}$; See [38] or Appendix C.

A Note on Notation. The plaintext space in Paillier is the ring of integers $\mathbb{Z}_N$ where N is a composite determined based on the security requirements of the system [38]. For visual simplicity in the presentation, additions, subtractions, and *scalar* multiplications (i.e., local computations) are left implicit unless otherwise stated, e.g., $[a] + [b]$, denotes the addition of two Paillier encryptions and $[a]c$ denotes the multiplication of an encrypted value by a public scalar.

Finally, several MPC protocols used require the *bit-length* of the encrypted value domain which we denote by ℓ. For example, $\ell = 64$ for representing 64-bit integers but can be higher for fixed-point or floating point representations.

Overview of Protocols:

`Reveal`$([a]) \rightarrow a$**.** Given an encryption $[a]$, obtain a in 1 round. This is realized by simply invoking the threshold-decryption protocol described in [11] and summarized in Appendix C of this work.

`Mult`$([a], [b]) \rightarrow [ab]$**.** Given encryptions $[a]$ and $[b]$, obtain the encryption $[ab]$ in 1 interactive round and 1 sub-protocol invocation. Details for instantiating this protocol are found in [12].

`TruncPR`$([a], \ell, m) \rightarrow [a/2^m]$**.** Given encryption $[a]$ and an integer m, obtain the encryption $[\tilde{a}] = [a/2^m]$. The protocol is statistically secure[3] and has a total complexity of 2 interactive rounds and $2m$ sub-protocol invocations.

`FPDiv`$([a], [b], \ell, f) \rightarrow [a/b]$**.** Given encryptions $[a]$ and $[b]$ obtain the encryption $[a/b]$ with f-bits of fixed-point precision. This protocol is based on the Goldschmidt method for achieving integer division presented in [21]. The idea behind Goldschmidt's method is to obtain initial approximations to both the divisor and dividend and iteratively compute *quadratically converging* approximations up to a desired precision. The MPC version of the protocol is described in [7], is statistically secure, and has a total complexity of $3\log_2(\ell) + 2$ interactive rounds and $1.5\ell \log_2(\ell) + 4\ell$ sub-protocol invocations.

5.3 Student's T-test

Student's T-test is used to compare means of two independent samples where the null hypothesis stipulates that there is no statistically significant difference between the two distributions [40].

Let π_1 and π_2 be two independent attributes (columns) in $\mathcal{D}$. Let x and y represent the column values of π_1 and π_2 in $\mathcal{D}$, respectively. That is, x is $(\mathcal{D}_{1,\pi_1}, \ldots, \mathcal{D}_{k,\pi_1})$ and y is $(\mathcal{D}_{1,\pi_2}, \ldots, \mathcal{D}_{k,\pi_2})$. Denote the mean of x and y as $\bar{x}$ and $\bar{y}$, respectively. Let the standard deviation of x and y be denoted as s_x and s_y.

The t statistic is computed according to:

$$t = \frac{\bar{x} - \bar{y}}{s_p \sqrt{\frac{2}{k}}} \tag{1}$$

Where $s_p = \sqrt{\frac{(k-1)s_x^2 + (k-1)s_y^2}{2k-2}}$ is an estimator of the pooled standard deviation of the two samples.

Computing Student's T-test in Custodes. Pseudo-code for the test is presented in Protocol 1. The protocol closely follows Equation 1 while adjusting the precision of fixed-point computations using `TruncPR` as explained in [7]. It avoids computing the square root of secret values and instead leaves it as a final operation to be computed on plaintext. We now briefly describe the steps of the protocol: Lines 1 to 3 compute the empirical mean of the two samples. Lines 4 to 11 compute the variance of the two samples. Lines 13 to 15 compute the square of the numerator in Equation 1. Line 16 computes the square of the denominator in Equation 1. Finally, line 19 computes the p-value (result) in the clear by using the square-root of the revealed t^2 value (test statistic) and the degrees of freedom, which, in the case of the Student's T-test is one less than the number of rows.

Protocol 1: $(\mathrm{t}, \mathrm{p}) \leftarrow$ `StudentTTest`$([\mathcal{D}], k, \pi_1, \pi_2)$

1 $([x], [y]) \leftarrow (\sum_{i=1}^{k} [\mathcal{D}_{i,\pi_1}], \sum_{i=1}^{k} [\mathcal{D}_{i,\pi_2}])$;
2 $[\bar{x}] \leftarrow$ `TruncPR`$([x]\mathtt{fp}_f(1/k), 2\ell, f)$;
3 $[\bar{y}] \leftarrow$ `TruncPR`$([y]\mathtt{fp}_f(1/k), 2\ell, f)$;
4 **foreach** $i \leftarrow 1, 2, \ldots, k$ **do parallel**
5 $\quad [d_{x_i}] \leftarrow [\mathcal{D}_{i,\pi_1}] - [\bar{x}]$;
6 $\quad [d_{y_i}] \leftarrow [\mathcal{D}_{i,\pi_2}] - [\bar{y}]$;
7 $\quad [h_{x_i}] \leftarrow$ `Mult`$([d_{x_i}], [d_{x_i}])$;
8 $\quad [h_{y_i}] \leftarrow$ `Mult`$([d_{y_i}], [d_{y_i}])$;
9 $([h_x], [h_y]) \leftarrow (\sum_{i=1}^{k} [h_{x_i}], \sum_{i=1}^{k} [h_{y_i}])$;
10 $[s_x] \leftarrow$ `TruncPR`$([h_x]\mathtt{fp}_f(1/(k-1)), 4\ell, 2f)$;
11 $[s_y] \leftarrow$ `TruncPR`$([h_y]\mathtt{fp}_f(1/(k-1)), 4\ell, 2f)$;
12 $[u] \leftarrow [s_x] + [s_y]$;
13 $[a] \leftarrow [\bar{x}] - [\bar{y}]$;
14 $[a^2] \leftarrow$ `Mult`$([a], [a])$;
15 $[a^2] \leftarrow$ `TruncPR`$([a^2], 2\ell, f)$;
16 $[b] \leftarrow$ `TruncPR`$([u]\mathtt{fp}_f(1/(k^2-k)), 2\ell, f)$;
17 $[t^2] \leftarrow$ `FPDiv`$([a^2], [b])$;
18 $t^2 \leftarrow$ `Reveal`$([t^2])$;
19 $\mathrm{p} \leftarrow \mathit{computePValue}(\sqrt{t^2}, k-1)$;
20 **return** $(\sqrt{t^2}, \mathrm{p})$;

[3] A definition of security–not to be confused with statistical tests. See Appendix A for details.

Requirements. Let $\eta \in \mathbb{N}$ be an upper bound on the largest absolute value in $\mathcal{D}$ (i.e., given as part of attributes' domain size), to evaluate a Student's T-test over any two attributes $\mathcal{D}$, the following constraints must hold to ensure no "overflow" occurs during computation: $\ell > \log_2(k) + 2\log_2(\eta) + f$ and $n > 2^{\ell+\kappa+n+1}$. The calculation is trivial to verifiy by examining the arithmetic circuit.

Complexity. All sub-protocols invoked are constant-rounds with the exception of FPDiv which requires a number of rounds proportional to $\log_2(\ell)$. Therefore, the total round complexity hinges on the FPDiv invocation making the final complexity $O(\log_2(\ell))$ rounds and $O(2k)$ invocations of constant-round sub-protocols.

5.4 Pearson Correlation Test

Pearson Correlation test is used to compare the linear correlation between two continuous independent variables. The result r lies in the range $[-1, 1]$ corresponding to negative or positive correlation level between the variables [40].

Let π_1 and π_2 be two continuous independent attributes. Let variables x and y represent the values of π_1 and π_2 where $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$ denote the observed values for x and y, respectively. Denote the mean of x and y as $\bar{x}$ and $\bar{y}$. Pearson Correlation's correlation coefficient r is then computed according to the following equation:

$$r = \frac{\sum_{i=1}^{k}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{k}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{k}(y_i - \bar{y})^2}} \tag{2}$$

Computing Pearson Correlation in Custodes. Pseudocode for computing Pearson Correlation coefficient is presented in Protocol 2. The protocol closely follows Equation 2 to compute the statistic and uses TruncPR to adjust the precision of fixed-point values. Lines 1 to 3 compute the empirical mean of the two samples. Lines 4 to 12 compute the sum of the variance of the two samples and the sum of the products of the standard deviations. Lines 13 to 16 compute the square of the numerator. Lines 17 to 18 compute the square of the denominator. Line 21 computes the p-value in the clear by using the square root of r^2 and degrees of freedom, which, in the case of the Pearson Correlation is $k - 2$.

Requirements. Let $\eta \in \mathbb{N}$ be an upper bound to the largest absolute value in $\mathcal{D}$, to evaluate a Pearson Correlation test over any two attributes in $\mathcal{D}$, the following requirements must hold for the user-set parameters to ensure no "overflow" occurs during computation: $\ell > 2\log_2(\eta) + \log_2(k) + f$ and $n > 2^{\ell+\kappa+n+1}$. Again, the calculation follows from the description of the circuit in the protocol.

Protocol 2: $(\mathsf{t}, \mathsf{p}) \leftarrow \mathtt{PearsonTest}([\mathcal{D}], k, \pi_1, \pi_2)$

1 $([x], [y]) \leftarrow (\sum_{i=1}^{k}[\mathcal{D}_{i,\pi_1}], \sum_{i=1}^{k}[\mathcal{D}_{i,\pi_2}])$;
2 $[\bar{x}] \leftarrow \mathtt{TruncPR}([x]\mathtt{fp}_f(1/k), 2\ell, f)$;
3 $[\bar{y}] \leftarrow \mathtt{TruncPR}([y]\mathtt{fp}_f(1/k), 2\ell, f)$;
4 **foreach** $i \leftarrow 1, 2, \ldots, k$ **do parallel**
5 $\quad [d_{x_i}] \leftarrow [\mathcal{D}_{i,\pi_1}] - [\bar{x}]$;
6 $\quad [d_{y_i}] \leftarrow [\mathcal{D}_{i,\pi_2}] - [\bar{y}]$;
7 $\quad [d^2_{x_i}] \leftarrow \mathtt{Mult}([d_{x_i}], [d_{x_i}])$;
8 $\quad [d^2_{y_i}] \leftarrow \mathtt{Mult}([d_{y_i}], [d_{y_i}])$;
9 $\quad [e_i] \leftarrow \mathtt{Mult}([d_{x_i}], [d_{y_i}])$;
10 $([s_x], [s_y]) \leftarrow (\sum_{i=1}^{k}[d^2_{x_i}], \sum_{i=1}^{k}[d^2_{y_i}])$;
11 $[s_x] \leftarrow \mathtt{TruncPR}([s_x], 2\ell, f)$;
12 $[s_y] \leftarrow \mathtt{TruncPR}([s_y], 2\ell, f)$;
13 $[a] \leftarrow \sum_{i=1}^{k}[e_i]$;
14 $[a] \leftarrow \mathtt{TruncPR}([a], 2\ell, f)$;
15 $[a^2] \leftarrow \mathtt{Mult}([a], [a])$;
16 $[a^2] \leftarrow \mathtt{TruncPR}([a^2], 2\ell, f)$;
17 $[b] \leftarrow \mathtt{Mult}([s_x], [s_y])$;
18 $[b] \leftarrow \mathtt{TruncPR}([b], 2\ell, f)$;
19 $[r^2] \leftarrow \mathtt{FPDiv}([a^2], [b])$;
20 $r^2 \leftarrow \mathtt{Reveal}([r^2])$;
21 $\mathsf{p} \leftarrow computePValue(\sqrt{r^2}, k-2)$;
22 **return** $(\sqrt{r^2}, \mathsf{p})$;

Complexity. All sub-protocols invoked are constant-rounds with the exception of FPDiv which requires a number of rounds proportional to $\log_2(\ell)$. We therefore conclude that the protocol requires $O(\log_2(\ell))$ rounds and 17 invocations.

5.5 Chi-Squared Test

The Chi-Squared test determines whether the sampling distribution of the test statistic follows a χ^2 distribution when the null hypothesis is true[9, 40]. The Chi-Squared test is useful in determining whether there is a significant difference between the expected frequencies and the observed frequencies in a set of observations from *mutually exclusive* categories. In other words, Chi-Squared evaluates the "goodness of fit" between a set of expected values and observed values, the test result is deemed significant if the expected frequencies match the observed frequencies.

For a collection of k observations classified into w mutually exclusive categories where each observed value is denoted by x_i for $i = 1, 2, \ldots, w$, denote the probability that a value falls into the ith category by q_i such that $\sum_{i=1}^{w} q_i = 1$. Note that the expected value for each category is $e_i = nq_i$. The Chi-Squared statistic is computed according to the following equation:

$$\chi^2 = \sum_{i=1}^{w} \frac{(x_i - e_i)^2}{e_i} \tag{3}$$

Let $\mathcal{D}$ contain w mutually exclusive attributes (categories) $\pi_1, \ldots, \pi_w$ such that $\mathcal{D}_i, \pi_j \in \{0, 1\}$ for all $i = 1 \ldots k$ and $j = 1 \ldots w$. In other words, each category is a Boolean flag representing whether a row i in $\mathcal{D}$ is in the category j. Given such a "raw" dataset $\mathcal{D}$, we need a way to convert $\mathcal{D}$ into histogram form $\mathcal{H} = (h_1, h_2, \ldots, h_w)$, filtered based on the selected categories, so as to compute the Chi-Squared statistic over $\mathcal{H}$. To achieve this in a private and secure manner, we must first "pre-process" $[\mathcal{D}]$ into a histogram $\mathcal{H}$ containing the summations of the w categories selected by the user.

Note: we use $\mathcal{H}$ for purpose of providing a general solution and to remain consistent with the descriptions of the previous two tests (i.e., $\mathcal{D}$ has the same format across all tests). If $\mathcal{D}$ is already in histogram form, the the Chi-Squared test may be applied directly on $\mathcal{D}$.

Chi-Squared in Custodes. Pseudo-code for computing the Chi-Squared test is presented in Protocol 3. It closely follows Equation 3. The first for-loop (line 1 computes the histogram $[\mathcal{H}] = ([h_1], \ldots, [h_w])$ from $[\mathcal{D}]$. The correctness of the computed histogram $\mathcal{H}$ follows from the fact that each attribute in the set is mutually exclusive, i.e., if $\mathcal{D}_{j,\pi_i} = 1$ then $\mathcal{D}_{j,\pi_w} = 0$ for all $j = 1 \ldots k$ and $i \neq w$. Line 3 computes the sum of all the values in $[\mathcal{H}]$. The second for-loop (line 4 computes the *expected values* of each entry in the $[\mathcal{H}]$ as well as the $(x_i - e_i)$ term of Equation 3. The third for-loop (line 7 computes each term in the summation. Finally, line 11 computes the Chi-Squared statistic by summing over all the individual terms. The computed statistic is used to compute the p-value using $(k-1)(w-1)$ for degrees of freedom.

Protocol 3: $(\mathsf{t}, \mathsf{p}) \leftarrow \mathtt{ChiSq}([\mathcal{D}], k, \{\pi_1, ..., \pi_w\}, \{q_1, ..., q_w\})$

1 **foreach** $i \leftarrow 1, 2, \ldots, w$ **do parallel**
2 $\quad [h_i] \leftarrow \sum_{j=1}^{k} [\mathcal{D}_{j,\pi_i}]$;
3 $[s] \leftarrow \sum_{i=1}^{w} [h_i]$;
4 **foreach** $i \leftarrow 1, 2, \ldots, w$ **do parallel**
5 $\quad [e_i] \leftarrow [s]q_i$;
6 $\quad [d_i] \leftarrow [h_i] - [e_i]$;
7 **foreach** $i \leftarrow 1, 2, \ldots, w$ **do parallel**
8 $\quad [w_i] \leftarrow \mathtt{Mult}([d_i], [d_i])$;
9 $\quad [w_i] \leftarrow \mathtt{TruncPR}([w_i], 2\ell, f)$;
10 $\quad [x_i] \leftarrow \mathtt{FPDiv}([w_i], [e_i])$;
11 $[\chi^2] \leftarrow \sum_{i=1}^{w} [x_i]$;
12 $\chi^2 \leftarrow \mathtt{Reveal}([\chi^2])$;
13 $\mathsf{p} \leftarrow computePValue(\chi^2, (k-1)(w-1))$;
14 **return** (χ^2, p)

Requirements. Let $\eta \in \mathbb{N}$ be an upper bound to the largest absolute value in $\mathcal{D}$. Let $\mathcal{H}$ be a histogram with w attributes $(2 \leq w \leq w)$. To correctly perform the Chi-Squared test over the w selected attributes in $\mathcal{D}$, the following requirements must hold for these user-set parameters to ensure no "overflow" occurs during computation: $\ell > \log_2(\eta) + 2\log_2(w) + f$ and $n > 2^{\ell+\kappa+n+1}$. The requirement follows from the test circuit description.

Complexity. All sub-protocols invoked are constant-rounds with the exception of `FPDiv` which requires a number of rounds proportional to $\log_2(\ell)$. We therefore conclude that the protocol requires $O(w \log_2(\ell))$ rounds and $3w + 1$ invocations.

6 SECURITY

We now argue that Custodes satisfies properties outlined in Section 3.

Correctness. Custodes follows the plaintext execution of statistical tests by using secure version of the sub-protocols as presented in Section 5. The main difference with plaintext execution of these tests is in the accuracy of the results since Custodes guarantees $O(f)$-bits of precision.

Confidentiality. The aim of Custodes is to enforce p-value calculation in a truthful manner. It achieves this goal by hiding the content of $\mathcal{D}$ and revealing only the metadata about the dataset $\mathcal{D}$ sufficient to carry out statistical tests and the results of the tests. The data owner reveals the following metadata:

- **The size of the dataset:** both the number of columns (attributes) and number of rows in each attribute, k and w, respectively. These values are necessary for both computing statistical tests and determining the associated p-values.
- **Attribute metadata:** information on the contents of $\mathcal{D}$ such as the characteristics of each attribute, the domain size, and independence from other attributes. For example, metadata for an "age" attribute may be the set `{AttrType: Age, NumericRange: 0-110}`). We stress, however, that the metadata can be made general and independent of the values in $\mathcal{D}$ when deemed of no influence on computation correctness.

The above information can be expressed as a function $\mathcal{L}_s : \mathbb{D}_{k \times w} \rightarrow \{0, 1\}^*$, that takes as input the dataset and returns k, w and other metadata.

We capture the confidentiality property of Custodes using a common experiment used in cryptography where an adversary is required to distinguish whether it is set in the real or an ideal world. In the real world, the adversary (i.e.,

a entity modeling the collusion of malicious parties) interacts with CUSTODES as it would in the real setting. In the ideal world, the adversary interacts with a simulator who has access only to the output of $\mathcal{L}_s(\mathcal{D})$ and to a test result oracle $O(\mathcal{D}, \cdot)$. The oracle $O(\mathcal{D}, \cdot)$ takes the database $\mathcal{D}$ and a statistical test circuit which it evaluates on $\mathcal{D}$ and returns its result. We note that O is a concept that is used only as part of the confidentiality definition and the proof thereof. In particular, it is used to capture the fact that the simulator is not given $\mathcal{D}$ but only the results of the tests. If one can show that an adversary cannot distinguish the real from the ideal world then CUSTODES does not reveal more about $\mathcal{D}$ than what the simulator knows about $\mathcal{D}$. Otherwise, the adversary could use this leaked information to distinguish between the two worlds. We formally capture this experiment below.

Definition 6.1. Let CUSTODES = (Setup, Compute, Audit), let $\mathcal{L}_s$ be a stateful metadata function, $O(\mathcal{D}, \cdot)$ a test result oracle, n is the number of parties and t is the threshold of parties required for decryption. Consider the following probabilistic experiments where $\mathcal{A}$ is an honest-but-curious, stateful adversary and $\mathcal{S}$ is a stateful simulator:

$\mathbf{Real}_{\mathcal{A}}(1^k, n, t)$: An adversary chooses $\mathcal{D}$ and a challenger C runs $\mathsf{Setup}(1^k, \mathcal{D}, n, t)$ and obtains the encrypted dataset $[\mathcal{D}]$ and keys $(\mathsf{pk}, \{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n\})$. *Wlog* the parties are split into two disjoint sets $\{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\}$ and $\{\mathcal{P}_t, \ldots, \mathcal{P}_n\}$ where the first set is controlled by $\mathcal{A}$ and the second by C who acts on behalf of honest parties. C sends $\mathcal{D}$, pk, and corresponding secret keys to all the parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$.

$\mathcal{A}$ then adaptively chooses q test queries $\{\mathcal{T}_1, \ldots, \mathcal{T}_q\}$ where each $\mathcal{T}_i$ corresponds to a valid statistical test circuit. For each $\mathcal{T}_i$, $\mathcal{A}$ and C engage in the Compute protocol that returns result t_i. In the end, $\mathcal{A}$ outputs a bit b.

$\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(1^k, n, t)$: An adversary chooses a dataset $\mathcal{D}$ and $\mathcal{S}$ is given the output of $\mathcal{L}_s(\mathcal{D})$. $\mathcal{S}$ runs $\mathsf{Setup}_{\mathcal{S}}(1^k, \mathcal{L}_s(\mathcal{D}), n, t)$ that returns an encryption of a random dataset $[\mathcal{D}_R]$ and $(\mathsf{pk}, \{\mathsf{sk}_1, \mathsf{sk}_2, \ldots, \mathsf{sk}_n\})$. $\mathcal{S}$ also updates its state. *Wlog* the parties are split into two disjoint sets $\{\mathcal{P}_1, \ldots, \mathcal{P}_{t-1}\}$ and $\{\mathcal{P}_t, \ldots, \mathcal{P}_n\}$ where the first set is controlled by $\mathcal{A}$ and the second by $\mathcal{S}$.

$\mathcal{A}$ then adaptively chooses q test queries $\{\mathcal{T}_1, \ldots, \mathcal{T}_q\}$ where each $\mathcal{T}_i$ corresponds to a valid statistical test circuit. For each $\mathcal{T}_i$, $\mathcal{S}$ calls $O(\mathcal{D}, \mathcal{T}_i)$ and gets the result t_i. $\mathcal{S}$ then calls $\mathsf{Compute}_{\mathcal{S}}(\mathsf{t}_i)$ where $\mathsf{Compute}_{\mathcal{S}}$ is an interactive protocol that simulates the interaction of the honest parties in the Compute phase with parties controlled by the adversary. In the end, $\mathcal{A}$ outputs a bit b.

We say that CUSTODES *is* $(\mathcal{L}_s, \mathsf{t}_1, \ldots, \mathsf{t}_q)$*-secure if* $\exists$ *PPT simulator* $\mathcal{S}$ *such that for all PPT adversaries* $\mathcal{A}$*,*

$$\left| \Pr[\mathbf{Real}_{\mathcal{A}}(1^k, n, t) = 1] \; - \; \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(1^k, n, t) = 1] \right| \leq \mathsf{negl}(k)$$

THEOREM 1. *CUSTODES is* $(\mathcal{L}_s, \mathsf{t}_1, \ldots, \mathsf{t}_q)$*-secure, i.e., no information beyond the metadata and the result of q statistical tests on* $\mathcal{D}$ *is revealed.*

See Appendix A for the proof of the above theorem.

Access Control. Though anyone can homomorphically compute on $[\mathcal{D}]$, the obtained encrypted result cannot be decrypted unless a threshold number of $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ collude. Moreover, each party performs a partial decryption of a result only if the decryption request message was posted on $\mathcal{B}$ and signed by one of the approved parties. Hence, decryption of a ciphertext can only occur if a party from $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ requested a decyrption.

Verifiability. This property is guaranteed again by the fact that a threshold number of the parties are required to decrypt a result. That is, even if someone computes on $[D]$, the obtained result is encrypted and will not be decrypted unless more than t of the parties collude. However, every test whose result (t, p) is decrypted has partial decryption shares of (t, p) recorded on $\mathcal{B}$. Hence, even if the party $\mathcal{P}$ that initiated the test goes offline after requesting a decryption of (t, p), the certificate on (t, p) can be reconstructed from shares stored on $\mathcal{B}$.

Auditability. Since all *local* computations are deterministic and the transcripts of *interactive protocols* are recorded on $\mathcal{B}$, given a certificate $\psi = (\tau, \mathcal{P}^{\mathsf{pk}}, \mathcal{T}, (\mathsf{t}, \mathsf{p}))$ an auditor $\mathcal{V}$ can evaluate the arithmetic circuit $\mathcal{T}([\mathcal{D}])$ *without interaction* and ensure that each input to a non-linear gate corresponds to a local computation of $\mathcal{T}([\mathcal{D}])$. Note that it suffices to ensure the input is correct given that the output is signed by all parties during the computation. We stress that an audit does not require any SMPC evaluations given that the transcript and (encrypted) result is available publicly on $\mathcal{B}$. Thus, an audit of a certificate ψ is essentially a linear scan of records obtained from $\mathcal{B}$ and local homomorphic ciphertext evaluations.

7 EXPERIMENTAL EVALUATION

We now turn to demonstrating the applicability of CUSTODES to the real world. The purpose of this section to understand the feasibility of CUSTODES for certifying hypotheses. The goals of our evaluation are as follows:

- Thoroughly evaluate CUSTODES on a variety of datasets in terms of overall runtime and dataset characteristics.

- Ensure correctness of statistical tests matches those of theoretical guarantees and compare the error (if any) to the equivalent computation performed outside of Custodes.
- Measure the impact of different selection parameters on the overall computation time for each statistical test.
- Evaluate the auditing process for statistical tests computed through Custodes.

7.1 Implementation and Environment

We implement Custodes in Go 1.10.1. The code is open-source and available at https://github.com/sachaservan/custodes. The implementation of statistical tests closely follows the pseudo-code described in Section 5. We implement the threshold-variant of the Paillier encryption scheme for performing the computation required for evaluating statistical tests but deviate slightly when computing division gates as we found that using linear-secret sharing methods for computing division was more efficient in practice. As such, we implement `FPDiv` protocol using Shamir secret sharing [39] and convert from *Paillier* ciphertext to shares using techniques from [11]. We stress, however, that this is only done for *two ciphertexts* (numberator and denominator of the division gate) and not the entire computation.

All experiments were conducted on a single machine with Intel Xeon E5 v3 (Haswell) @ 2.30GHz processors (40 cores in total) and 36GB of RAM, running Ubuntu 16.04 LTS. Unless otherwise stated, each result is the average over five separate runs (for each combination of parameters). While in most cases the variance in runtime across runs was minimal, we nonetheless report 95%-confidence markers (under a normal approximation assumption).

7.2 Datasets

We evaluate each statistical test on synthetic and real-world datasets. Notice, however, that Custodes's performance is not impacted by the distribution of the underlying data since the data itself is encrypted. We nonetheless evaluate Custodes on two real-world datasets to illustrate this fact.

The real-world datasets are obtained from the UC Irvine Machine Learning Repository [1]. For experiments involving Student's T-test and Pearson Correlation test we use the Abalone dataset containing weights and heights of 4177 Tasmanian abalones. For Chi-Squared we use the Pittsburgh Bridges dataset which contains categorical data on 108 bridges in the city of Pittsburgh.

In addition to these datasets, we generate three continuous synthetic datasets containing 1,000, 5,000 and 10,000 rows, respectively, with random real value entries ranging between 1 and 100. We use these datasets to evaluate the Student's T-test and the Pearson Correlation test. Again, we stress that generating datasets at random does not advantage Custodes in any way because computations are distribution agnostic. For evaluating Chi-Squared, we generate three categorical datasets containing 20 mutually exclusive categories. We evaluate the Chi-Squared test on a subset of 5, 10 and 20 categories to demonstrate the impact of varying the number of selected attributes. The characteristics of the datasets used in our evaluation is summarized in Table 2.

Dataset	Size	Attributes	Type	Range
Abalone	4177	2	Continuous	[0, 1.37]
Pittsburgh	108	4	Categorical	{0, 1}
cat_1k	1000	5, 10, 20	Categorical	{0, 1}
cat_5k	5000	5, 10, 20	Categorical	{0, 1}
cat_10k	10000	5, 10, 20	Categorical	{0, 1}
rand_1k	1000	2	Continuous	[0, 100.0]
rand_5k	5000	2	Continuous	[0, 100.0]
rand_10k	10000	2	Continuous	[0, 100.0]

Table 2: Dataset characteristics

7.3 Experiment Setup

We benchmark all statistical tests with 3 computing parties (servers), where a threshold majority of ($t = 2$) are required to partcipate in order to decrypt (i.e., at least one server is honest and does not attempt to collude). Each party is run as a separate process and with access to two cores on the computing machine. In other words, each party is simulated as a dual-core machine, separate from other parties[4]. This setup allows us to simulate a multi-party computing environment while simultaneously controlling for all external variables, (e.g., network latency) in the system. We set the simulated network latency (delay between communication rounds) to 1ms which simulates average latencies observed on a local area network (LAN); a standard setup used for benchmarking multi-party computations [4, 44, 47].

7.4 Parameters

To ensure all tests are evaluated in a way that enables comparisons between results, we fix the parameters of Custodes ahead of time to satisfy the requirements imposed by all three statistical tests (and datasets) and do not change the parameters between experiments. Specifically, we set $f = 30$ which provides approximately 9 decimal places of precision for the Chi-Squared test and 5 decimal places for Student's T-test and Pearson Correlation test (due to the square-root computation

[4]The machine on which the experiments are conducted has 40 cores which allows such a setup without CPU swapping.

	Mean Absolute Error
Student's T-test	3.40×10^{-10}
Pearson Correlation Test	6.92×10^{-7}
Chi-Squared Test	2.35×10^{-9}

Table 3: Absolute error of statistical tests computed through Custodes compared to the equivalent test computed using the SciPy package.

at the end). We let the statistical security parameter $\kappa = 40$ which provides 40-bits of statistical security; a common default value [4]. The largest value found in all the datasets is 100.0 so we set $\eta_{max} = 100.0$ and the number of entries in the largest dataset is $k_{max} = 10,000$. We fix $\ell = 100$ which ensures $\ell > 4\log_2(\eta_{max}) + 4\log(k_{max}) + f$ thus satisfying parameter requirements for all three statistical tests. Finally, we set N (the Paillier modulus) to be a 1024-bit composite which both ensures security of ciphertexts and guarantees that $\mathbb{Z}_N$ (the encryption space) is large enough to support statistical test computations with the given parameters.

7.5 Results

We break down the results into four sections each of which roughly corresponds to a goal that we outline at the beginning of this section.

System Setup. Across all experiments, the amount of time required to setup Custodes (i.e., generating keys, encrypting the dataset, etc) remained below 1 second with mean 0.30 seconds, sd. 0.28. This affirms that the overhead placed on the data owner is minimal.

Correctness. We compare the resulting precision of each statistical test computed in Custodes with results obtained from computing the same test over the same data using the Python SciPy Library [30]. Table 3 reports the mean errors per test. The results did not vary significantly from the mean. The absolute error for all tests was between 0.0 and 0.0009. We observe that the empirical error obtained in Custodes matches the precision expected based on the parameter selection and the probabilistic correctness of the `TruncPR` protocol.

Runtime Evaluation. We run each statistical test on each configuration of parameters five times and report the average runtime in Table 4. The Chi-Squared test had the highest maximum runtime at 1m40s, with mean. Both Student's T-test and Pearson Correlation had a maximum runtime of approximately 2m10s.

In all experiments, division required the most computation time which is not unexpected considering that it is the most computationally expensive protocol invoked by each test. Perhaps surprisingly, the number of parties involved has a larger impact on performance than the size of the dataset. This is due to the high overhead of computing division over encrypted data which requires many interactions between parties. Both Protocol 1 and 2, require only a single division operation making the computation independent of the dataset size. Therefore, for significantly larger datasets this relationship is not likely to hold true. However, when computing the Chi-Squared test, division is overwhelmingly the dominant contributor to the overall runtime. This is due to the nature of the Chi-Squared statistic which *requires one division per category* (in the general case) which equates to a total of 20 calls to `FPDiv` for the largest selection of categories used in the Chi-Squared experiments. In practice, however, certain assumptions (e.g., when the expectation is uniform across categories) make it possible to evaluate the Chi-Squared test with only a single division operation (for the purpose of obtaining the reciprocal) and would thus considerably reduce the computing overhead.

Auditing. For each test computed in our evaluation, we store the locally computed ciphertext trace in addition to the results of interactive protocols which we then use to evaluate the audit process for each computed statistical test. As expected, the audit verified successfully across all computed tests. The total audit time was consistently below 10 seconds with mean 2.48, sd. 2.34. This demonstrates that while computing statistical tests in Custodes incurs a computational overhead, the auditing process remains relatively efficient.

Regarding Scalability. The results of the runtime evaluation begs the question: can Custodes scale to a setting with more than a handful of computing parties? We believe the answer is yes. We note that in the experimental evaluation we set the threshold of parties necessary for decryption and computations to be a *majority* of the total number of parties in the system. In practice, however, this is too stringent a requirement. From a technical standpoint, a system with n parties need only have t of them present in the computation phase and t need not scale with n. Therefore, even in a setting with thousands of researchers, similar performance can be achieved as demonstrated in our evaluation.

8 RELATED WORK

Much of the related work surrounding Custodes either focuses on private computations using homomorphic encryption or non-cryptographic methods for preventing p-hacking such as methods for pre-registration of hypotheses. To our knowledge, there has been no prior work using cryptographic techniques for certifying the validity of statistical tests in an auditable way.

Chi-Squared Test Runtime Evaluation				
Dataset	**Runtime; 4 categories**	**Runtime; 5 categories**	**Runtime; 10 categories**	**Runtime; 20 categories**
Pittsburgh	24069ms (std. 1458ms)	–	–	–
cat_1k	–	27890ms (std. 836ms)	47365ms (std. 1614ms)	97192ms (std. 5801ms)
cat_5k	–	27700ms (std. 2034ms)	48860ms (std. 1765ms)	97412ms (std. 5794ms)
cat_10k	–	28671ms (std. 1905ms)	50413ms (std. 2057ms)	98683ms (std. 3325ms)

Pearson Correlation Test Runtime Evaluation	
Dataset	**Runtime**
Abalone	62795ms (std. 12880ms)
rand_1k	2967ms (std. 1426ms)
rand_5k	7030ms (std. 13610ms)
rand_10k	133095ms (std. 23905ms)

Student's T-test Test Runtime Evaluation	
Dataset	**Runtime**
Abalone	42993ms (std. 7392ms)
rand_1k	2317ms (std. 1800ms)
rand_5k	6238ms (std. 9166ms)
rand_10k	100310ms (std. 16783ms)

Table 4: Runtime comparisons between datasets and statistical tests for 2-out-of-3 threshold scheme.

Lauter *et al.* [34] and Zhang *et al.*[43] propose the use of homomorphic encryption to compute statistical tests on encrypted genomic data. However, the constructions are not intended for validating statistical testing procedures but rather for guarding the privacy of patient data. Furthermore, Lauter *et al.* make several simplifications such as not performing encrypted division (rather performing arithmetic division in the clear), imposing assumptions on the data, etc., making their solution less general compared to methods for computing statistical tests in Custodes.

Homomorphic encryption and multi-party computation techniques have been used for outsourcing machine learning tasks of private data [5, 22, 41, 42]. Our work, however, crucially relies on the decryption functionality being decentralized in order to control and keep track of data exposure.

Several *non-cryptographic* techniques exist to guard against the MCP in statistical analysis. For example, there are several procedures that adjust p-values in scenarios where multiple hypotheses are examined at once. These range from conservative protocols that bound the family-wise error rate [15] to more relaxed procedures that bound the ratio of false rejections among the rejected tests (false discovery rate, FDR) [3] or the marginal FDR [18, 45]. When applied properly these techniques prevent p-hacking in an idealized scenario. However, there is no guarantee that they are applied correctly. A research can misuse these procedures, intentional or unintentional, and only apply them over a subset of all statistical tests being computed on a given dataset. Furthermore, there is no way for external auditors to verify how these methods were applied. We, on the other hand, leverage these techniques by operating on encrypted data and tracking statistical tests in a tamper-proof ledger which certifies the correctness of results in an auditable way.

Dwork *et al.* [16, 17] propose a method that constrains analyst's access to a hold-out dataset as follows. A trusted party keeps a hold-out dataset and answers up to m hypotheses using a differentially private algorithm. Here, m depends on the size of the hold-out data and the generalization error that one is willing to tolerate. Hence, compared to our decentralized approach, this method assumes *trust into a single party* that (1) does not release the dataset to the researchers and (2) does not answer more than m hypotheses queries.

Finally, we note that preserving privacy of dataset values when releasing results of statistical tests [19, 28] is orthogonal to our work.

9 CONCLUSIONS

We present Custodes, a system that certifies hypothesis testing using proven cryptographic techniques and a decentralized certifying authority. Custodes computes statistical test over encrypted data and uses blockchain technology to provide auditability of those results. We believe that Custodes is a viable solution to prevent p-hacking and control for false-discoveries in a way that promotes accountability and reporducability in scientific studies. We discuss various theoretical constructions of Custodes and provide details on our particular implementation in which we support three common statistical tests. We evaluate this implementation on various configurations and dataset sizes.

ACKNOWLEDGMENTS

We would like to thank Anna Lysyanskaya for providing us with editorial feedback and design suggestion as well as Andy van Dam for his support.

REFERENCES

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[2] C. G. Begley and L. M. Ellis. Drug development: Raise standards for preclinical cancer research. *Nature*, 483(7391):531, 2012.

[3] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

[4] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security*, pages 192–206. Springer, 2008.

[5] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[6] O. Catrina and S. De Hoogh. Improved primitives for secure multiparty integer computation. In *International Conference on Security and Cryptography for Networks*, pages 182–199. Springer, 2010.

[7] O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *International Conference on Financial Cryptography and Data Security*, pages 35–50. Springer, 2010.

[8] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, and E. Shi. Solidus: Confidential distributed ledger transactions via PVORM. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 701–717, 2017.

[9] W. G. Cochran. The $\chi 2$ test of goodness of fit. *The Annals of Mathematical Statistics*, pages 315–345, 1952.

[10] A. Cockburn, C. Gutwin, and A. Dix. Hark no more: on the preregistration of chi experiments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 141. ACM, 2018.

[11] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 280–300. Springer, 2001.

[12] I. Damgård, M. Jurik, and J. B. Nielsen. A generalization of Paillier's public-key system with applications to electronic voting. *International Journal of Information Security*, 9(6):371–385, 2010.

[13] K. Dickersin, S. Chan, T. Chalmersx, H. Sacks, and H. Smith Jr. Publication bias and clinical trials. *Controlled clinical trials*, 8(4):343–353, 1987.

[14] Y. Dodge. *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008.

[15] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[16] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Guilt-free data reuse. *Communications of the ACM*, 60(4):86–93, 2017.

[17] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. L. Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126. ACM, 2015.

[18] D. P. Foster and R. A. Stine. α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(2):429–444, 2008.

[19] M. Gaboardi, H. Lim, R. Rogers, and S. Vadhan. Differentially private chi-squared hypothesis testing: Goodness of fit and independence testing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2111–2120, 2016.

[20] M. M. Ghassemi, S. E. Richter, I. M. Eche, T. W. Chen, J. Danziger, and L. A. Celi. A data-driven approach to optimized medication dosing: a focus on heparin. *Intensive care medicine*, 40(9):1332–1339, 2014.

[21] R. E. Goldschmidt. *Applications of division by convergence*. PhD thesis, Massachusetts Institute of Technology, 1964.

[22] T. Graepel, K. Lauter, and M. Naehrig. ML confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology (ICISC)*, 2013.

[23] M. L. Head, L. Holman, R. Lanfear, A. T. Kahn, and M. D. Jennions. The extent and consequences of p-hacking in science. *PLoS biology*, 13(3):e1002106, 2015.

[24] K. E. Henry, D. N. Hager, P. J. Pronovost, and S. Saria. A targeted real-time early warning score (trewscore) for septic shock. *Science translational medicine*, 7(299):299ra122–299ra122, 2015.

[25] J. P. Ioannidis. Contradicted and initially stronger effects in highly cited clinical research. *Jama*, 294(2):218–228, 2005.

[26] J. P. Ioannidis. Why most published research findings are false. *PLoS medicine*, 2(8):e124, 2005.

[27] L. K. John, G. Loewenstein, and D. Prelec. Measuring the prevalence of questionable research practices with incentives for truth telling. *Psychological science*, 23(5):524–532, 2012.

[28] A. Johnson and V. Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1079–1087, 2013.

[29] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[30] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2018-07-01].

[31] M. G. Kendall et al. The advanced theory of statistics. *The advanced theory of statistics.*, (2nd Ed), 1946.

[32] N. L. Kerr. Harking: Hypothesizing after the results are known. *Personality and Social Psychology Review*, 2(3):196–217, 1998.

[33] L. Lamport. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[34] K. Lauter, A. López-Alt, and M. Naehrig. Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer, 2014.

[35] L. Mayaud, P. S. Lai, G. D. Clifford, L. Tarassenko, L. A. G. Celi, and D. Annane. Dynamic data during hypotensive episode improves mortality predictions among patients with sepsis and hypotension. *Critical care medicine*, 41(4):954, 2013.

[36] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. http://www.bitcoin.org/bitcoin.pdf.

[37] U. S. Neill. Publish or perish, but at what cost? *The Journal of clinical investigation*, 118(7):2368–2368, 2008.

[38] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[39] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[40] T. H. Wonnacott and R. J. Wonnacott. *Introductory statistics*, volume 5. Wiley New York, 1990.

[41] D. Wu and J. Haven. Using homomorphic encryption for large scale statistical analysis. Technical report, Technical Report: cs. stanford. edu/people/dwu4/papers/FHESI Report. pdf, 2012.

[42] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. E. Lauter, and M. Naehrig. Crypto-nets: Neural networks over encrypted data. *CoRR*, abs/1412.6181, 2014.

[43] Y. Zhang, W. Dai, X. Jiang, H. Xiong, and S. Wang. Foresee: Fully outsourced secure genome study based on homomorphic encryption. 15:S5, 12 2015.

[44] Y. Zhang, A. Steele, and M. Blanton. Picco: a general-purpose compiler for private distributed computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 813–826. ACM, 2013.

[45] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management*

of Data, pages 527–540. ACM, 2017.

[46] J. Zhou, D. Foster, R. Stine, and L. Ungar. Streaming feature selection using alpha-investing. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 384–393. ACM, 2005.

[47] G. Zyskind et al. *Efficient secure computation enabled by blockchain technology*. PhD thesis, Massachusetts Institute of Technology, 2016.

A DEFINITIONS AND PROOFS

Statistical Security. We make extensive use of statistically MPC protocols introduced in [6, 7]. The advantage of using statistical security (as opposed to perfect) is that many existing protocols can be rendered far more efficient with only slightly more relaxed notions of security.

Definition A.1 (Statistical Security [7])**.** For any two random variables X and Y with finite sample spaces U and V, respectively. The statistical distance between X and Y is said to be *statistically indistinguishable* in the security parameter κ if:

$$\Delta(X,Y) = \frac{1}{2}\sum_{w \in U \cup V} |\Pr[X = w] - \Pr[Y = w]| \leq \mathtt{negl}(\kappa)$$

In the special case that $\Delta(X,Y) = 0$ we say the distributions variables are *perfectly indistinguishable* or *information-theoretic secure.*

We refer the reader to [6] for a full description of statistically secure building blocks as well the proof of security. We note that when using threshold-Paillier as the backbone for MPC (as we do in Custodes), the security assumptions are reduced to a *computationally bounded* adversary and thus provides weaker security guarantees than provided by ideal instantiations of statistically secure protocols.

Proof of Theorem 1. Recall the claim of Theorem 1: Custodes is $(\mathcal{L}_s, t_1, \ldots, t_q)$-secure.

Proof. In order to prove the above statement, we need to instantiate $\text{Setup}_{\mathcal{S}}$ and $\text{Compute}_{\mathcal{S}}$ and show that they can produce interactions of the honest parties that are indistinguishable from those in Custodes' protocol. $\text{Setup}_{\mathcal{S}}$ is straightforward: $\mathcal{S}$ creates a random database of the same size as $\mathcal{D}$ since it is given this information as part of $\mathcal{L}_s(\mathcal{D})$. It then calls key generation of the encryption protocol.

In order to simulate the interactions during the test computation, $\mathcal{S}$.Compute is instantiated by combining simulators of the Custodes subprotocols outlined in Section 5.2 that are secure and universally composable. Finally, in order to reveal the correct test result, $\mathcal{S}$ uses the decryption simulator and the test result it obtained from the test result oracle $O(\mathcal{D}, \cdot)$. □

B ADDITIONAL EXPERIMENTS

Interactive Protocol Complexity

We measure the total number of multiplications required per statistical test and combination of parameters. Since `Mult` is the fundamental building block of *all* interactive protocols, this measurement provides an estimate for the total number of interactive rounds required between parties for computing a given test. We report the results in Table 5 and 6).

Student's T-test and Pearson Correlation Complexity		
Dataset	**Student's T-test**	**Pearson Correlation**
Abalone	35,969	40,147
rand_1k	29,615	30,616
rand_5k	37,615	42,616
rand_10k	47,615	57,616

Table 5: Number of multiplications required to compute a Student's T-test and Pearson Correlation test for each dataset.

Chi-Squared Test Interactive Complexity			
Dataset	**5 categories**	**10 categories**	**20 categories**
Pittsburgh	N/A	N/A	N/A
cat_1k	138,105	276,180	552,330
cat_5k	138,105	276,180	552,330
cat_10k	138,105	276,180	552,330

Table 6: Number of multiplication required to compute a Chi-Squared test for each dataset.

For both Student's T-test and Pearson Correlation test the number of multiplications ranged between 35,969 and 57,516 and was dependent on the total number of rows in the dataset. We stress that this also includes the computation of the division gate and therefore the number of multiplications does not scale linearly with the number of rows for this pair of tests (given there is only a single division operation).

For Chi-Squared test, the number of multiplications is independent of the number of rows. This is expected given that the protocol for computing Chi-Squared is dependent only on the number of categories selected and not on the number of observations in the dataset.

C ADDITIONAL TECHNICAL DETAILS

Threshold-Paillier Scheme

Let $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a set of n parties where a threshold majority are required to participate in order to decrypt (reveal) a ciphertext.

`KeyGen`$(k) \rightarrow (\text{pk}, \{\text{sk}_1, \ldots, \text{sk}_n\})$. Pick two k-bit primes, p and q such that $p = 2p' + 1$ and $q = 2q' + 1$ where p', q'

are Sophie Germain primes. Set $N = pq$, $M = p'q'$, $\lambda = LCM(p-1, q-1)$. Pick an integer d such that $d \equiv 1 \pmod{N^2}$ and $d \equiv 0 \pmod{M}$. [5] Set t such that $t \leq n$ and construct the polynomial $f(X) = \sum_{i=0}^{t-1} a_i X^i \pmod{N^2M}$ where a_i for $i \geq 1$ is picked at random from the set $\{0, 1, \ldots, N^2M - 1\}$. Set $a_0 = d$ and for $i = 1 \ldots n$, $\mathsf{sk}_i = f(i)$ corresponds ith share of the secret key $\mathsf{sk} = \lambda$ such that any subset of t can be used to decrypt a ciphertext.
Set the public key $\mathsf{pk} = N$.
Output $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.

Enc(pk**,** $m \in \mathbb{Z}_N$**)** $\rightarrow c \in \mathbb{Z}_{N^2}$**.** To encrypt a message $m \in \mathbb{Z}_N$, pick a random $r \in \mathbb{Z}_{N^2}$ and compute the ciphertext $c = (N+1)^m r^{N^2} \pmod{N^2}$. Output c.

ThresDec($c \in \mathbb{Z}_{N^2}, \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$**)** $\rightarrow m \in \mathbb{Z}_N$**.** To decrypt a ciphertext c, each player $\mathcal{P}_i \in \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ uses the private secret key share sk_i to compute $c_i = c^{2\Delta \mathsf{sk}_i}$ where $\Delta = n!$. With the required t unique shares $(c_1, \ldots, c_t)$, the decryption can be computed by taking the set $C = \{c_1, \ldots, c_t\}$ of t shares and combining them using Lagrange interpolation as follows:

$$c' = \prod_{i \in C} c_i^{2\lambda^C_{0,i}} \text{ where } \lambda^C_{0,i} = \Delta \prod_{i' \in C \setminus \{i\}} \frac{i}{i'-i} \in \mathbb{Z}$$

Note that $c' = c^{4\Delta^2 f(0)} = c^{4\Delta^2 d}$. Since $4\Delta^2 d \equiv 0 \pmod{\lambda}$ and $4\Delta^2 d \equiv 4\Delta^2 d \pmod{N^2}$, we get that $c' = (n+1)^{4\Delta^2 m}$ where m is the plaintext we wish to extract. Therefore, $4\Delta^2 m$ can be (efficiently) obtained as described in the original Paillier which allows us to recover m as follows:

$$m = L(c' \bmod N^2)(4\Delta^2)^{-1} \pmod{N}$$

where $L(u) = \frac{u-1}{N}$ as specified in [12, 38]. Output m.

EvalAdd($c_1, c_2 \in \mathbb{Z}_{N^2}$**)** $\rightarrow c' \in \mathbb{Z}_{N^2}$**.** Addition of two encrypted values $c_1, c_2 \in Z_{N^2}$ is computed as follows: pick a random $r \in \mathbb{Z}_{N^2}$ and compute $c' = (c_1 c_2) r^{N^2} \pmod{N^2}$. Observe that $c_1 c_2 = (N+1)^{m_1+m_2}(r')^{N^2}$ for some $r' \in \mathbb{Z}_{N^2}$ hence the result is equivalent to encrypting $m_1 + m_2$ using r' for randomness making the result correct. We note that the randomization step may be omitted when deterministic computations are acceptable. Output c'.

EvalMult($c \in \mathbb{Z}_{N^2}$**,** $b \in \mathbb{Z}_N$**)** $\rightarrow c' \in \mathbb{Z}_{N^2}$**.** Multiplication of a ciphertext $c \in Z_{N^2}$ by a public constant $b \in \mathbb{Z}_N$ is computed as follows: pick a random $r \in \mathbb{Z}_{N^2}$ and compute $c' = c^b r^{N^2} \pmod{N^2}$. Observe that $c^b = (N+1)^{mb}(r')^{N^2}$ for some $r' \in \mathbb{Z}_{N^2}$, hence the result is correct. Again, we emphasize that the randomization step may be omitted when needed to achieve deterministic computations. Output c'.

[5] [12] warns the reader that using $d = \lambda$ (as per the original Paillier scheme) is not secure in the threshold setting. Therefore, d must be chosen independently of λ.

Fixed-Point Arithmetic in $\mathbb{Z}_N$

Representation of real numbers in both Paillier and secret sharing schemes is a common problem given the message space consists of elements from $\mathbb{Z}_N$, respectively. Performing arithmetic over real numbers is therefore non-trivial and requires either floating-point or fixed-point encoding. While floating-point representation provides better precision, it is also far less efficient compared to fixed-point representation, at least in an SMC context [47].

Encoding. Given a public fixed-point precision parameter f denoting bits of precision required per computation, we can approximate any real number $a \in \mathbb{R}$ as $a \approx \lfloor a2^f \rfloor 2^{-f}$. We define $\mathsf{fp}_f(a) = \lfloor a2^f \rfloor$ to be the function encoding real numbers into the corresponding fixed-point representation.

Linear Arithmetic. Addition (and subtraction) of two fixed-point numbers is trivial and can be computed without interaction. Let a, b be two fixed-point numbers with f-bits of precision, then $a + b = (a+b)2^{-f}$ with f-bits of precision, as required.

Multiplication. Multiplication of fixed-point numbers is more involved as we need to *scale down* the result to have the correct precision. For two fixed-point numbers a, b with f-bits of precision, observe that $ab = ab2^{-2f}$ has $2f$-bits of precision. Hence, we need to *scale down* the product ab by 2^f in order to obtain the desired precision. This can be achieved interactively using the `TruncPR` protocol to obtain $(ab)/2^f$ so that the resulting value has f-bits of precision as required.

Signed Number Encoding in $\mathbb{Z}_N$

We designate a range of length ω (e.g., $\omega = N$) within $\mathbb{Z}_N$ for the representation of negative integers. More concretely we let the range $[0, \lceil \frac{\omega}{2} \rceil)$ represent the *positive* integers in the range $[0, \lceil \frac{\omega}{2} \rceil)$ and elements in the range $[\lceil \frac{\omega}{2} \rceil, \omega)$ encode *negative* integers in the range $[-\lceil \frac{\omega}{2} \rceil, 0)$. Note that the correctness of both addition and multiplication is preserved with such encoding allowing for efficient representation and arithmetic of signed integer values in $\mathbb{Z}_N$ assuming the domain of encrypted values is the range $[-\lceil \frac{\omega}{2} \rceil, \lceil \frac{\omega}{2} \rceil]$.