Secure Federated Correlation Test and Entropy Estimation

Qi Pang [*1†] **Lun Wang** [*2‡] **Shuai Wang** [3] **Wenting Zheng** [1] **Dawn Song** [4]

Abstract

We propose the first federated correlation test framework compatible with secure aggregation, namely FED-χ^2. In our protocol, the statistical computations are recast as frequency moment estimation problems, where the clients collaboratively generate a shared projection matrix and then use stable projection to encode the local information in a compact vector. As such encodings can be linearly aggregated, secure aggregation can be applied to conceal the individual updates. We formally establish the security guarantee of FED-χ^2 by proving that only the minimum necessary information (i.e., the correlation statistics) is revealed to the server. We show that our protocol can be naturally extended to estimate other statistics that can be recast as frequency moment estimations. By accommodating Shannon'e Entropy in FED-χ^2, we further propose the first secure federated entropy estimation protocol, FED-H. The evaluation results demonstrate that FED-χ^2 and FED-H achieve good performance with small client-side computation overhead in several real-world case studies.

1. Introduction

Correlation test, as the name implies, is the process of examining the correlation between two random variables using observational data. It is a fundamental building block in a wide variety of real-world applications, including feature selection (Zheng et al., 2004), cryptanalysis (Nyberg, 2001), causal graph discovery (Spirtes et al., 2000), empirical finance (Ledoit & Wolf, 2008; Kim & Ji, 2015), medical studies (Kassirer, 1983) and genomics (Wilson et al., 1999; Dudoit et al., 2003). Entropy estimation also has a wide application like clustering (Rosenberg & Hirschberg, 2007), complexity measurement (Harrison, 1992), DNN compression (Park et al., 2017), anomaly detection (Nychis et al., 2008), etc. Because the observational data used in the correlation tests and entropy estimation may contain sensitive information such as genomic information, and collecting participants' information to a central repository poses a significant privacy risk. To address this problem, we utilize the federated setting, where each client maintains its own data and communicates with a central server to calculate a function. The communication transcript should contain as little information as feasible to prevent the server from inferring sensitive information.

To motivate our work and ease the understanding of the problem setting, we consider a medical company that wants to study the correlation between genetic defects and races using the patients' private data from several hospitals. For a traditional method in the federated setting, the server, which is the medical company, will aggregate the hospitals' local private contingency tables[1] using secure aggregation (Bonawitz et al., 2017; Bell et al., 2020). The company can conduct correlation tests with the aggregated global contingency table without directly accessing the individual hospitals' private data. Attentive readers might be aware that the method mentioned above leaks the joint distribution, which is the whole global contingency table, to the server. The joint distribution may contain sensitive information, and leaking it will probably violate privacy regulations. For instance, the medical company can observe the genetic distribution across races from the global table.

The secure aggregation primarily supports linear aggregation. However, in correlation tests and entropy estimation, the calculations involve computing a summed p-th moment and summed logarithmic function over the aggregated data, where $p \in (0,1) \cup (1,2]$. Thus, the joint distribution will be leaked if we directly apply secure aggregation. To bridge the gap between secure aggregation and the important applications like federated correlation tests and entropy estimation, we take an important step towards designing nonlinear secure aggregation protocols. Specifically, we design

[*]Equal contribution [1]Carnegie Mellon University [2]Google [3]Hong Kong University of Science and Technology [4]UC Berkeley. Correspondence to: Qi Pang <qpang@andrew.cmu.edu>, Lun Wang <lunwang@google.com>.

[†]Part of the work is done when Qi Pang was at HKUST. [‡]Work is done when Lun Wang was at UC Berkeley.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]Contingency table contains the frequency distribution of the variables; see (Wikipedia, 2021).

federated protocol frameworks, namely FED-χ^2 and FED-H, optimized for χ^2-test and Shannon's Entropy. FED-χ^2 and FED-H are designed to *have low computation and communication costs* and *only disclose information that is much less sensitive than the joint distribution*. Our first insight is to recast correlation tests and entropy estimation as frequency moment estimation problems. To approximate the frequency moments in a federated manner, each client collaborates with the other clients to generate a projection matrix and encodes its raw data into a low-dimensional vector via stable random projection (Indyk, 2006; Vempala, 2005; Li, 2008). Such encodings can be aggregated with only summation, allowing clients to leverage secure aggregation to aggregate the encodings. The server then decodes the aggregated encoding to approximate the frequency moments. As secure aggregation conceals each client's individual update within the aggregated global update, the server learns only necessary information for the correlation test and entropy estimation.

We evaluate FED-χ^2 on 4 synthetic datasets and 16 real-world datasets. The results show that FED-χ^2 can replace centralized correlation tests with good accuracy. Compared to the traditional method with secure aggregation mentioned above, FED-χ^2 saves a factor of $\mathcal{O}(m)$ communication cost per client, where m is the size of the contingency tables. In FED-χ^2, clients only need to upload a low-dimensional encoding with size $\ell \ll m$, while in the traditional method the clients will upload the complete contingency tables. Additionally, we analyze FED-χ^2 in two real-world use cases: feature selection and online false discovery rate control. The results show that FED-χ^2 can achieve comparable performance with centralized correlation tests and can withstand up to 20% of clients dropping out with only minor influence on the accuracy. Besides Pearson's χ^2-test, we also demonstrate how to estimate the entropy in federated setting by accommodating Shannon's Entropy in FED-χ^2. We refer to this protocol as FED-H, and we also empirically show the effectiveness of FED-H.

In summary, we make the following contributions:

- We propose FED-χ^2, the first secure federated correlation test framework, which is computation- and communication-efficient and leaks much less information than directly using secure aggregation to collect the contingency table.
- FED-χ^2 decomposes correlation test into frequency moments estimation that can easily be encoded/decoded using stable projection and secure aggregation. We provide formal security proof and utility analysis of the protocol.
- We demonstrate how to accommodate other statistical computations like Shannon's Entropy in FED-χ^2 and propose the first secure federated entropy estimation protocol FED-H.
- We empirically evaluate FED-χ^2 and FED-H in several real-world use cases. The findings suggest that FED-χ^2 and FED-H can substitute centralized χ^2-test and Shannon's Entropy estimation with comparable accuracy.

We maintain our code at https://github.com/Qi-Pang/Federated-Correlation-Test.

2. Related Work

There have been a line of works studying secure federated learning or statistics. Bonawitz et al. (2017) proposed the well-quoted secure aggregation protocol as a low-cost way to securely calculate linear functions in a federated setting. It has seen many variants and improvements since then. For instance, Truex et al. (2019) and Xu et al. (2019) employed advanced crypto tools for secure aggregation, such as threshold homomorphic encryption and functional encryption. So et al. (2021) proposed TURBOAGG, which combines secure sharing with erasure codes for better dropout tolerance. To address the class imbalance issue in federated learning, Wang et al. (2021b) proposed a monitoring scheme that can infer the composition of training data for each FL round. Zhu et al. (2023) studied the robust mean estimators with optimal statistical rates to defend against malicious clients. To improve communication efficiency, Bell et al. (2020) and Choi et al. (2020a) replaced the complete graph in secure aggregation with either a sparse random graph or a low-degree graph.

Secure aggregation is deployed in a variety of applications. Agarwal et al. (2018) added binomial noise to local gradients, resulting in both differential privacy and communication efficiency. Wang et al. (2020) replaced the binomial noise with discrete Gaussian noise, which is shown to exhibit better composability. Kairouz et al. (2021) proved that the sum of discrete Gaussian is close to discrete Gaussian, thus discarding the common random seed assumption from Wang et al. (2020).

The above three studies incorporate secure aggregation in their protocols to lower the noise scale required for differential privacy. Chen et al. (2020) added an extra public parameter, ensuring uniform training and enabling malicious clients detection during aggregation. Choi et al. (2020b) proposed using count-sketch with secure aggregation for frequency estimation. Rothchild et al. (2020) applied count-sketch in federated learning to improve communication. Chen et al. (2022) investigated the communication cost of differentially private federated learning, and optimized it using count-sketch. For entropy estima-

tion, Zhao et al. (2007) estimated the entropy of origin-destination flows using L_p norm estimation.

Nevertheless, designing secure federated correlation tests and entropy estimations, despite their importance in real-world scenarios, are not explored by existing research in this field.

3. Methodology

In this section, we elaborate on the design of FED-χ^2 and FED-H. Sec. 3.1 formalizes the problem, establishes the notation system, and introduces the threat model. In Sec. 3.2, we detail the design of FED-χ^2. In Sec. 3.3 and 3.4, we present the security proof, utility analysis, communication and computation analysis of FED-χ^2. In Sec. 3.5, we show how Shannon's Entropy estimation can be accommodated in FED-χ^2 and propose FED-H.

3.1. Problem Setup

We first formulate the problem of the federated correlation test and establish the notation system, and later we will extend it to federated entropy estimation. We use $[n]$ to denote $\{1, \cdots, n\}$. We denote vectors with bold lower-case letters (*e.g.*, $\mathbf{a}, \mathbf{b}, \mathbf{c}$) and matrices with bold upper-case letters (*e.g.*, $\mathbf{A}, \mathbf{B}, \mathbf{C}$).

For the ease of representation, we use the example we mentioned in Sec. 1 to introduce all the notations. A medical company is studying the correlation between genetic defects (denoted by variable X) and race (denoted by variable Y). The support domain of X (or Y) is denoted by $\mathcal{X}$ (or $\mathcal{Y}$). In the example, $\mathcal{X} = \{yes, no\}$ representing whether the participant has the genetic defect, and $\mathcal{Y}$ is the set of all races. We denote the size of $\mathcal{X}$ as m_x, the size of $\mathcal{Y}$ as m_y.

The company wants to use the patient records from n hospitals to conduct the research. Concretely, each hospital holds a 2-dimensional local contingency table $\mathcal{D}_i = \{x \in \mathcal{X}, y \in \mathcal{Y} : v_{xy}^{(i)} \in \{0\} \cup [M]\}$, where x is the row label, y is the column label, and $v_{xy}^{(i)}$ is the number of patients with the label (x, y). We use $m = m_x m_y$ to denote the size of the contingency table.

The first step of the traditional method in federated setting is to collect all the hospitals' contingency tables on a centralized server $\mathcal{S}$ of the company and sum them to obtain the global contingency table $\mathcal{D} = \{x, y : v_{xy} = \sum_{i\in[n]} v_{xy}^{(i)}\}$. The total number of samples with row label x (or column label y) is defined as $v_x = \sum_{y\in\mathcal{Y}} v_{xy}$ (or $v_y = \sum_{x\in\mathcal{X}} v_{xy}$). The total number of samples observed is $v = \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} v_{xy}$.

The next step is to calculate a test statistic, $s(\mathcal{D})$, on the global table. For Pearson's χ^2-test, the statistic is as below:

$$s_{\chi^2}(\mathcal{D}) := \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \frac{(v_{xy} - \bar{v}_{xy})^2}{\bar{v}_{xy}}, \tag{1}$$

where $\bar{v}_{xy} = \frac{v_x \times v_y}{v}$ is the expectation of v_{xy} if X and Y are uncorrelated. The statistics is then compared with a threshold to decide whether X and Y are correlated.

Attentive readers might be aware that the method described above incurs severe ethical issues that the patient records from different hospitals are collected on a centralized server of the company, which probably violates corresponding privacy regulations. In this work, our aim is to design a secure federated correlation test protocol only leaking non-sensitive information with low computation/communication cost. Concretely, we trade off accuracy for security, as long as the estimation error is small with a high probability. Formally, if FED-χ^2 outputs $\hat{s}$, whose corresponding standard centralized correlation test output is s, the following accuracy requirement should be satisfied with small multiplicative error bound ϵ and small failure probability δ:

$$\mathbb{P}[(1-\epsilon)s \le \hat{s} \le (1+\epsilon)s] \ge 1 - \delta \tag{2}$$

Threat Model. We assume that the centralized server $\mathcal{S}$ is honest but curious. It honestly follows the protocol due to regulatory or reputational pressure but is curious to discover extra private information from clients' legitimate updates for profit or surveillance purposes. As a result, the clients updates should contain as little sensitive information as feasible.

On the other hand, we assume the clients (*e.g.* the hospitals) are honest and won't collude with the server. Specifically, we do not consider client-side adversarial attacks (*e.g.*, data poisoning attacks (Bagdasaryan et al., 2020; Bhagoji et al., 2019)). However, we allow a small portion of clients to drop out during the execution. We also provide further security analysis when collusion between the server and the client happens in Appendix H.

More importantly, we assume that the marginal distributions of the variables are not sensitive while the joint distribution is. The above example is a natural case where such an assumption holds. The aggregated marginal distributions of the genetic defects and the races won't leak sensitive information. However, the correlation between a specific pair of race and genetic defect can be easily observed if the joint distribution, which is the aggregated global contingency table, is obtained by the server.

3.2. Secure Federated Correlation Test

In this section, we introduce the design of FED-χ^2 in detail. We also discuss how the design generalizes to other statistical computations such as Shannon's Entropy (Shannon, 1948) in Sec. 3.5.

From Federated Correlation Test to Frequency Moments Estimation. The α-th frequency moment of a key-value stream is formally defined as below:

Definition 1 (α-th frequency moment)**.** *Given a key-value stream $\{a_t \in \mathcal{A}, b_t \in \mathcal{B}\}_{t\in[T]}$, the α-th frequency moment of $\mathcal{S}$ is defined as:*

$$\mathbb{F}_\alpha(\mathcal{S}) := \sum_{a\in\mathcal{A}} \Big(\sum_{t\in[T]:a_t=a} b_t \Big)^\alpha \tag{3}$$

We observe that the test statistics of many correlation tests can be rewritten as frequency moments. For example, the statistic of χ^2-test can be reformatted as a second frequency moment:

$$s_{\chi^2}(\mathcal{D}) = \sum_{x,y} \frac{(v_{xy} - \bar{v}_{xy})^2}{\bar{v}_{xy}} = \sum_{x,y} \Big(\frac{v_{xy} - \bar{v}_{xy}}{\sqrt{\bar{v}_{xy}}} \Big)^2 \tag{4}$$

In the federated setting, the ith client calculates the vector $\mathbf{u}_i(x,y) := \frac{v^{(i)}_{xy} - \bar{v}_{xy}/n}{\sqrt{\bar{v}_{xy}}}$, and the above formula can be rewritten as a second frequency moment estimation problem:

$$s_{\chi^2}(\mathcal{D}) = \sum_{x,y} \Big(\frac{v_{xy} - \bar{v}_{xy}}{\sqrt{\bar{v}_{xy}}} \Big)^2 = \sum_{x,y} \Big(\sum_{i\in[n]} \mathbf{u}_i(x,y) \Big)^2 \tag{5}$$

Federated Frequency Moments Estimation. Now that we have reformatted the problem, the second step is to design the messages transmitted in FED-χ^2 for α^{th} frequency moments estimation. We use *stable projection* (Indyk, 2006; Vempala, 2005) to encode the client-side information and *geometric mean estimator* (Li, 2008) to decode the aggregated message. Before we dive into the details, let's refresh some preliminaries. See Appendix A for more details on stable distribution.

Definition 2 (Symmetric α-stable distribution)**.** *A random variable X follows a symmetric α-stable distribution $\mathcal{Q}_{\alpha,\beta,F}$ if its characteristic function is as follows:*

$$\phi_X(t) = \exp(-F|t|^\alpha(1 - \sqrt{-1}\beta \operatorname{sgn}(t) \tan(\frac{\pi\alpha}{2}))), \tag{6}$$

where F is the scale, $\alpha^{th} \in (0,2]$ is the stability parameter, and β is the skewness.

α-stable distribution is named due to its property called α-stability. Briefly, the sum of independent α-stable variables still follows an α-stable distribution with a different scale.

Definition 3 (α-stability)**.** *If random variables $X \sim \mathcal{Q}_{\alpha,\beta,1}, Y \sim \mathcal{Q}_{\alpha,\beta,1}$ and X and Y are independent, then $C_1X + C_2Y \sim \mathcal{Q}_{\alpha,\beta,C_1^\alpha+C_2^\alpha}$.*

Inspired by the idea of Indyk's well-cited paper (Indyk, 2006), we encode the frequency moments in the scale parameter of a stable distribution. To encode information contained in the local contingency table $\mathcal{D}_i$, the ith client collaborates with other clients to generate a projection matrix $\mathbf{P} \in \mathbb{R}^{\ell\times m}$ projection matrix, where ℓ is the encoding size. The components of $\mathbf{P}$ are drawn independently from an α-stable distribution $\mathcal{Q}_{\alpha,0,1}$. The client then calculates $\mathbf{u}_i$ as defined in Eq. 5 and applies the projection get $\mathbf{e}_i := \mathbf{P} \times \mathbf{u}_i$ as the encoding (lines 1-2 in Alg. 1).

To decode, the server first sums the encodings from all the clients $\mathbf{e} := \sum_{i\in[n]} \mathbf{e}_i$. According to the α-stability defined in Definition 3, every component e_k in the encoding vector $\mathbf{e}$, $k \in [\ell]$, follows this stable distribution $\mathcal{Q}_{\alpha,0,s(\mathcal{D})}$. Thus, the statistic of the correlation test can be estimated with the scale of the distribution. We estimate the scale using an unbiased geometric mean estimator (Li, 2008) (lines 3-4 in Alg. 1).

A significant advantage of stable projection is that the encodings are linearly aggregatable and thus compatible with secure aggregation. Secure aggregation only reveals the aggregated encoding to the server and greatly reduces the privacy leakage. Furthermore, in Sec. 3.4, we show that a small encoding size suffices to accurately approximate the frequency moments with high probability and can potentially improve communication cost with certain setups.

FED-χ^2 Protocol. We present the complete FED-χ^2 protocol in Alg. 2. Firstly, the marginal statistics v_x, v_y and v are collected with secure aggregation and broadcasted to all the clients (lines 1–6 of Alg. 2). This step can be omitted if the marginal statistics are already known. The ith client calculates $\mathbf{u}_i$ (lines 9–10 of Alg. 2), and samples a random seed r_i and broadcasts to other clients (line 11 of Alg. 2). Then, the clients receive the random seeds and sample the projection matrix $\mathbf{P}$ from the α-stable distribution $\mathcal{Q}^{\ell\times m}_{2,0,1}$ using the common random seed r (lines 12–13 of Alg. 2). The ith client projects $\mathbf{u}_i$ to obtain the encoding $\mathbf{e}_i$ (line 14 of Alg. 2). Then, the encodings are quantized and aggregated with secure aggregation (line 15 of Alg. 2). As we have already known the marginal statistics in the first round, the quantization bound can be set accordingly. Additionally, we can use high precision for quantization, such as 64 bits, such that the precision of the quantized float numbers is comparable to or even better than the IEEE floating numbers. We validate this conjecture with empirical evaluation and hence ignore the effect of quantization on accuracy in the analysis. In the last step, the server gets the χ^2-test statistics using the decoding algorithm described in Alg. 1 (line 17 of Alg. 2).

Remark: Client Dropout. Attentive readers might ask what if some clients drop out during the protocol execution? We argue that dropouts in the first round have no effect on the test's accuracy as long as the secure aggregation used is resilient to dropout, such as (Bonawitz et al., 2017; Bell et al., 2020). On the other hand, dropouts in the second

Algorithm 1 The encoding and decoding scheme (Indyk, 2006) for federated frequency moments estimation. Note that the encoding and decoding themselves do not provide security guarantee.

Function ENCODE $(\boldsymbol{P}, \boldsymbol{u}_i)$:
  return $\mathbf{P} \times \mathbf{u}_i$
Function DECODE $(\boldsymbol{e})$:
  return $\frac{\prod_{k=1}^{\ell} |\mathbf{e}_k|^{2/\ell}}{(\frac{2}{\pi}\Gamma(\frac{2}{\ell})\Gamma(1-\frac{1}{\ell})\sin(\frac{\pi}{\ell}))^{\ell}}$ // ℓ is the encoding size.

Algorithm 2 The complete FED-χ^2 protocol. SECUREAGG is a remote procedure that receives inputs from the clients and returns the summation to the server. INITSECUREAGG is the corresponding setup protocol deciding the communication graph and other hyper-parameters.

Round 1: Reveal the marginal statistics
  INITSECUREAGG (n) // n: clients number
  for $x \in [m_x]$ **do** $v_x = \text{SECUREAGG}(\{v_x^{(i)}\}_{i\in[n]})$
  for $y \in [m_y]$ **do** $v_y = \text{SECUREAGG}(\{v_y^{(i)}\}_{i\in[n]})$
  Server
    Calculate $v = \sum_x v_x$ and Broadcast v, $\{v_x\}$, and $\{v_y\}$ to all the clients
Round 2: Approximate the statistics
  Client $i \in [n]$
    Calculate $\bar{v}_{xy} = \frac{v_x v_y}{v}$
    Prepare $\mathbf{u}_i$ s.t. $\mathbf{u}_i(x, y) = \frac{v_{xy}^{(i)} - \bar{v}_{xy}/n}{\sqrt{\bar{v}_{xy}}}$
    Randomly sample a random seed r_i and broadcast to all the other clients
    Collect the random seeds from the other clients and obtain the shared random seed $r = \sum_i r_i$
    Sample the projection matrix $\mathbf{P}$ from $\mathcal{Q}_{2,0,1}^{\ell\times m}$ using the common random seed r
    Calculate $\mathbf{e}_i = \text{ENCODE}(\boldsymbol{P}, \boldsymbol{u}_i)$
  $\mathbf{e} = \text{SECUREAGG}(\text{QUANTIZE}(\{\boldsymbol{e}_i\}_{i\in[n]}))$
  Server
    $\hat{s}_{\chi^2} = \text{DECODE}(\boldsymbol{e})$

round will affect the accuracy of the test. However, since the χ^2 value is typically far from the decision threshold, FED-χ^2 is intrinsically robust to a small portion of clients dropping out (see Section 4 for empirical assessment).

Remark: The Selection of Secure Aggregation. As introduced in Sec. 2, there are a variety of secure aggregation protocols for different setups (Bonawitz et al., 2017; Truex et al., 2019; Xu et al., 2019; So et al., 2021; Bell et al., 2020; Choi et al., 2020a). In the rest of the paper, we choose the state-of-the-art cross-device secure aggregation protocol by (Bell et al., 2020) due to its simple trust assumption and low communication cost. We want to emphasize that our protocols can incorporate any secure aggregation protocols as needed.

3.3. Security Analysis

We now prove the security enforced by Alg. 2 via a standard simulation proof process (Lindell, 2017) on the basis of Theorem 1.

Theorem 1 (Security)**.** *Let Π be an instantiation of Alg. 2 with the secure aggregation protocol in Alg. 4 of Appendix B with cryprographic security parameter λ. There exists a PPT simulator* SIM *such that for all clients $\mathcal{C}$, the number of clients n, all the marginal distributions $\boldsymbol{v}_x, \boldsymbol{v}_y$, and the aggregated encoding $\boldsymbol{e}$, the output of* SIM *is indistinguishable from the view of the real server $\Pi_{\mathcal{C}}$ in that execution,* i.e., $\Pi_{\mathcal{C}} \approx_{\lambda} \text{SIM}(\boldsymbol{e}, \boldsymbol{v}_x, \boldsymbol{v}_y, n)$.

Intuitively, Theorem 1 illustrates that no more information about the clients except the aggregated updates is revealed to the centralized server. Note that this is the minimal necessary information for the server to estimate the test statistic. The complete proof for Theorem 1 is in Appendix D.

To further emphasize the privacy protection of our protocol, we also provide analysis on the leakage when the server colludes with a client in Appendix H. We show that even the collusion happens, our protocol can still successfully hide the information in a subspace with exponential possible distributions, which practically enforce privacy given the considerably large size of the solution space.

3.4. Utility, Communication & Computation Analysis

We first present the utility analysis of FED-χ^2 in Alg. 2. We show that the output of FED-χ^2, $\hat{s}_{\chi^2}$, is a fairly accurate approximation (parameterized by ϵ) to the correlation test output s_{χ^2} in the standard centralized setting with high probability parameterized by δ when ℓ is appropriately chosen. The proof is deferred to Appendix E.

Theorem 2 (Utility)**.** *Let Π be an instantiation of Alg. 2 with secure aggregation protocol in Alg. 4 of Appendix B. Π is parameterized with $\ell = \frac{1}{\eta_{\alpha,\epsilon}} \log(2/\delta)$, where $\eta_{\alpha,\epsilon}$ is a function of α and ϵ with $\lim_{\epsilon\to 0} \eta_{\alpha,\epsilon} \to 0$. After executing $\Pi_{\mathcal{C}}$ on all clients $\mathcal{C}$, the server yields $\hat{s}_{\chi^2}$, whose distance to the accurate correlation test output s_{χ^2} is bounded with high probability as follows:*

$$\mathbb{P}[\hat{s}_{\chi^2} < (1-\epsilon)s_{\chi^2} \vee \hat{s}_{\chi^2} > (1+\epsilon)s_{\chi^2}] \leq \delta \tag{7}$$

Then we present the communication and computation cost of Alg. 2.

Theorem 3 (Communication Cost)**.** *Let Π be an instantiation of Alg. 2 with secure aggregation protocol in Alg. 4 of Appendix B, then (1) the client-side communication cost is $\mathcal{O}(\log n + m_x + m_y + \ell)$; (2) the server-side communication cost is $\mathcal{O}(n \log n + nm_x + nm_y + n\ell)$.*

Theorem 4 (Computation Cost)**.** *Let Π be an instantiation of Alg. 2 with secure aggregation protocol in Alg. 4*

of Appendix B, then (1) the client-side computation cost is $\mathcal{O}(\log^2 n + (\ell + m_x + m_y)\log n + m\ell)$; (2) the server-side computation cost is $\mathcal{O}(n\log^2 n + n(\ell + m_x + m_y)\log n + \ell)$.

Note that compared with the original computation cost presented in (Bell et al., 2020), the client-side overhead has an extra $\mathcal{O}(m\ell)$ term. This term is incurred by the encoding overhead. We also give an empirical evaluation on the client-side computation overhead in Sec. 4.1. Please refer to Appendix F for the detailed proof of Theorem 3 and Theorem 4.

3.5. Secure Federated Entropy Estimation

FED-χ^2 provides a practical way to estimate the non-linear aggregation results without first leaking the joint distribution. Specifically, FED-χ^2 computes the second order frequency moments of the aggregated data without leaking additional information. We deem that our framework is not limited to computing the second order frequency moments, instead, by generating projection matrix $\mathbf{P} \sim \mathcal{Q}_{\alpha,0,1}$, any α-order frequency moments, $\alpha \in (0,1) \cup (1,2]$, can be efficiently and securely estimated in federated setting.

FED-H Protocol. In the following section, we will show how to accommodate Shannon's Entropy in FED-χ^2 and form our protocol of FED-H. Similar to the setting of federated correlation test, in federated entropy estimation, each client i holds the observed data $v_x^{(i)}$, $x \in \mathcal{X}, i \in [n]$. n is the number of the clients and the size of $\mathcal{X}$ is denoted as m_x. Similarly, $v_x = \sum_{i=1}^n v_x^{(i)}$ and we assume that the data is the frequency and normalized: $\sum_{x\in\mathcal{X}} v_x = 1$. Then, the Shannon's Entropy is:

$$H = -\sum_{x\in\mathcal{X}} v_x \log v_x \tag{8}$$

To compute H in the federated setting of our framework, we recast Eq. 8 as two p-order frequency moments, which is also adopted in (Zhao et al., 2007):

$$H = -\sum_{x\in\mathcal{X}} v_x \log v_x = -\frac{1}{2\Delta}\Big(\sum_{x\in\mathcal{X}} v_x^{1+\Delta} - \sum_{x\in\mathcal{X}} v_x^{1-\Delta}\Big), \tag{9}$$

where Δ is a small consent value. In this way, we recast Shannon's Entropy as two frequency moment estimation of orders $1+\Delta$ and $1-\Delta$. The rest of the protocol is the same as FED-χ^2 in Alg. 2 except that we estimate two frequency moments. The security proof and utility analysis can also be generalized to FED-H naturally.

Utility Analysis. In the previous sections, we have shown that using α-stable projection to estimate v_x^α, the following holds w.h.p.:

$$\widehat{\sum_{x\in\mathcal{X}} v_x^{1+\Delta}} \in (1\pm\epsilon)\sum_{x\in\mathcal{X}} v_x^{1+\Delta},\ \widehat{\sum_{x\in\mathcal{X}} v_x^{1-\Delta}} \in (1\pm\epsilon)\sum_{x\in\mathcal{X}} v_x^{1-\Delta} \tag{10}$$

Given that the frequencies $v_x \le 1$ and by union bound, w.h.p., we have:

$$\hat{H} = -\frac{1}{2\Delta}\Big(\widehat{\sum_{x\in\mathcal{X}} v_x^{1+\Delta}} - \widehat{\sum_{x\in\mathcal{X}} v_x^{1-\Delta}}\Big) \in H \pm \Omega(\epsilon) \tag{11}$$

We prove Eq. 11 in Appendix G. If we use properly large encoding size ℓ when estimating $v_x^{1+\Delta}$ and $v_x^{1-\Delta}$ to produce small ϵ, the additive error in FED-H will be negligible as we will show in our evaluation in Sec. 4.3. The security, communication, and computation analysis in Sec. 3.3 and Sec. 3.4 directly applies to FED-H.

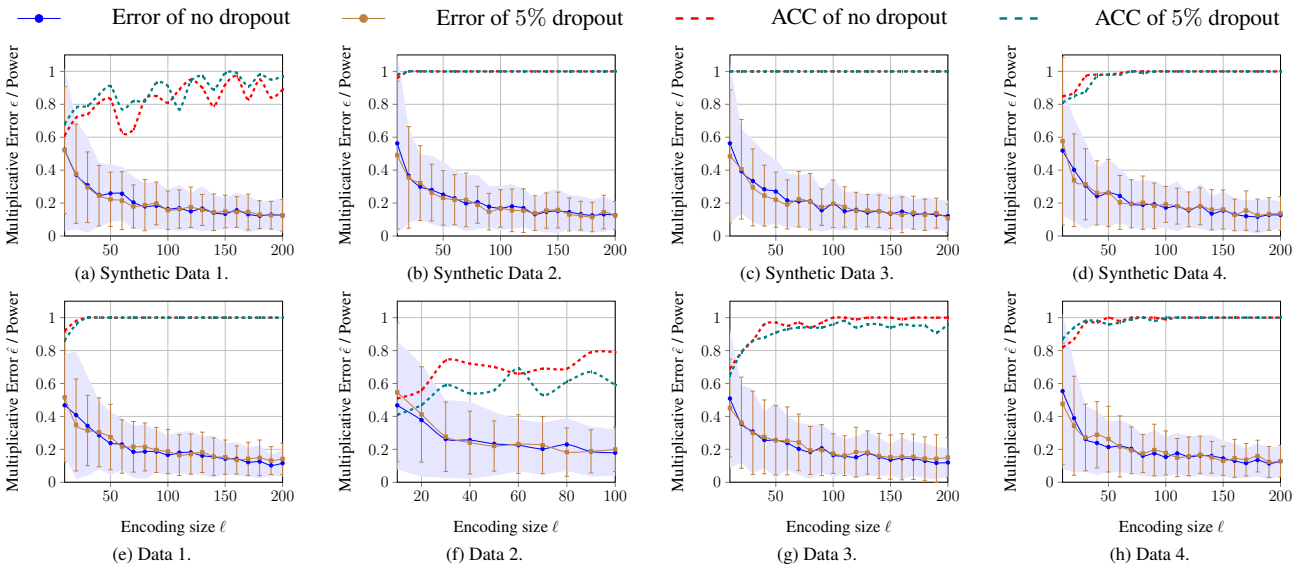
4. Evaluation

In this section, we would like to empirically answer the following questions: (1) How much does the accuracy decrease due to the lossy encoding? How does the accuracy loss change with the size of encoding l? (2) How long does the encoding take at the client side? (3) Is the performance overhead small enough in real-world applications? We primarily evaluate the performance of FED-χ^2 and then empirically verify the effectiveness of FED-H.

Experiment Setup. To assess FED-χ^2's accuracy, we simulate it on four synthetic datasets and 12 real-world datasets in Sec. 4.1. We compare the multiplicative error $\hat{\epsilon} := \frac{|\hat{s}_{\chi^2}(\mathcal{D}) - s_{\chi^2}(\mathcal{D})|}{s_{\chi^2}(\mathcal{D})}$ and power of FED-χ^2 with that of the standard centralized χ^2-test. The four synthetic datasets are independent, linearly correlated, quadratically correlated, and logistically correlated. For the real-world datasets, we report the details in Appendix I. We also evaluate FED-χ^2's utility in two real-world application scenarios in Sec. 4.2: feature selection and online false discovery rate (FDR) control. For feature selection, we report the model accuracy trained on the selected features. For online FDR control, we report the average false discovery rate. We compare the performance of FED-χ^2 with that of the centralized χ^2-test in each of the two experiments.

Similar to FED-χ^2, to verify the effectiveness of FED-H, we simulate it on four synthetic datasets and compare the additive error $\hat{\epsilon} := |\hat{H}(\mathcal{D}) - H(\mathcal{D})|$ in Sec. 4.3. The four synthetic datasets are also independent, linearly correlated, quadratically correlated, and logistically correlated, and they are normalized as mentioned in Sec. 3.5.

For secure aggregation, we discretize all the real numbers to 64-bit fix-point numbers. We provide further evaluation on the influence of finite field size in Appendix O, which shows that FED-χ^2 is numerically stable under different finite field sizes.



(a) Synthetic Data 1. (b) Synthetic Data 2. (c) Synthetic Data 3. (d) Synthetic Data 4.

(e) Data 1. (f) Data 2. (g) Data 3. (h) Data 4.

Figure 1: Multiplicative error and power of FED-χ^2 w.r.t. encoding size ℓ with and without dropout.

Unless otherwise specified, experiments are launched on an Ubuntu 18.04 LTS server equipped with 32 AMD Opteron(TM) Processor 6212 and 512GB RAM.

4.1. Performance of FED-χ^2

Accuracy. We begin by evaluating the accuracy of FED-χ^2, as illustrated in Fig. 1. Each point represents the mean of 100 independent runs with 100 clients, while the error bars indicate the standard deviation. We choose $m_x = m_y = 20$ in this experiment. Note that the accuracy drop of FED-χ^2 is independent of the number of clients. For the full version of Fig. 1, please refer to Appendix J.

From Fig. 1, we observe that the larger the encoding size ℓ, the smaller the multiplicative error. When $\ell = 50$, the multiplicative error $\epsilon \approx 0.2$.

We also evaluate the power (Cohen, 2013) of FED-χ^2. We set the p-value threshold as 0.05. From the dashed lines in Fig. 1, we can tell that the power of FED-χ^2 is high. This conforms with our observation on the multiplicative errors. Since the χ^2 values are typically far from the decision threshold, a multiplicative error of 0.2 rarely flips the final decision.

We also present the results when 5% of clients drop out in the second round of FED-χ^2 in Fig. 1, which show that FED-χ^2 is robust to a small portion of dropouts. In Appendix L, we present the results in terms of 10%, 15%, and 20% dropout rates. The results further show that FED-χ^2 can tolerate a considerable portion of clients dropout.

Client-side Computation Overhead. To assess extra computation overhead incurred by FED-χ^2 on the client side, we measure the execution time of the encoding scheme on an Android 10 mobile device equipped with a Snapdragon865 CPU and 12GB RAM. We use PyDroid (Sandeep Nandal, 2020) to run the client-side computation of FED-χ^2 on the Android device.

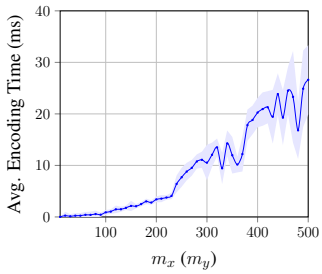


Figure 2: Client-side encoding overhead.

As shown in Fig. 2, each point represents the average of 100 separate runs, with accompanying error bars. The overhead is generally negligible. For example, for a 500×500 contingency table, the encoding takes less than 30ms. The overhead grows linearly in relation to m_x (m_y) and consequently quadratically in Fig. 2, where $m_x = m_y$.

4.2. Downstream Use Case Study

Feature Selection. Our first case study explores secure federated feature selection using FED-χ^2. The setting is that each client holds data with a large feature space and wants to collaborate with other clients to rule out unimportant features and retain features with top-k highest χ^2 scores. We use Reuters-21578 (Hayes & Weinstein, 1990), a standard text categorization dataset (Yang, 1999; Yang & Pedersen, 1997; Zhang & Yang, 2003), and pick the top-20 most frequent categories using 17,262 training and 4,316 test doc-

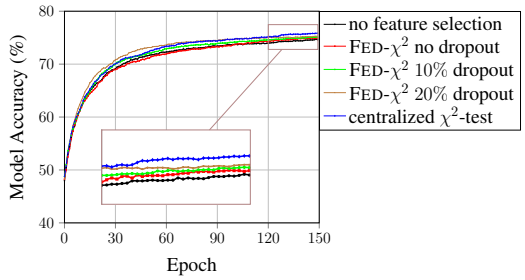


Figure 3: Accuracy of models trained with features selected by FED-χ^2 and centralized χ^2-test.

uments. These documents are distributed randomly to 100 clients, each of whom receives the same number of training documents. After removing all numbers and stop-words, we obtain 167,135 indexing terms.

The contingency table is of size 2×20 where 2 corresponds to whether a term occurs in an article and 20 corresponds to the number of different article categories. After performing feature selection using FED-χ^2, we select the top 40,000 terms with the highest χ^2 scores. When compared with the centralized χ^2-test, 38,012 (95.03%) of the selected terms are identical, indicating that FED-χ^2 produces highly consistent results with the standard χ^2-test.

We then train logistic regression models using the terms selected by FED-χ^2 and the centralized χ^2-test, respectively. All hyper-parameters are the same. The details of these models are reported in Appendix K. The training and testing splits are the same for FED-χ^2, centralized χ^2-test, and model without feature selection (i.e. there are 17,262 training and 4,316 test documents). We use the same learning rate; random seed and all other settings are also the same to make the comparison fair. We get the result of Fig. 3 and the models are all trained on NVIDIA GeForce RTX 3090.

The results in Fig. 3 further demonstrate that FED-χ^2 exhibits comparable performance with the centralized χ^2-test. When 10% and 20% of clients dropout in the second round of FED-χ^2, the accuracy of the trained model using the features selected by FED-χ^2 does not drop much. We also examine performance without feature selection, and as expected, model accuracy is significantly greater after feature selection. Note that the model without feature selection has 2,542,700 more parameters than the model with feature selection. Hence, feature selection effectively improves model accuracy while reducing model size and computational cost. We also provide further evaluation on the influence of encoding size ℓ in Appendix N, which shows that FED-χ^2 can achieve comparable performance with the centralized χ^2-test under different ℓ.

Online False Discovery Rate Control. In the third case study, we explore federated online false discovery rate

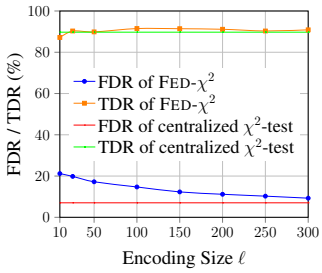


Figure 4: FDR & TDR w.r.t. ℓ for SAFFRON.

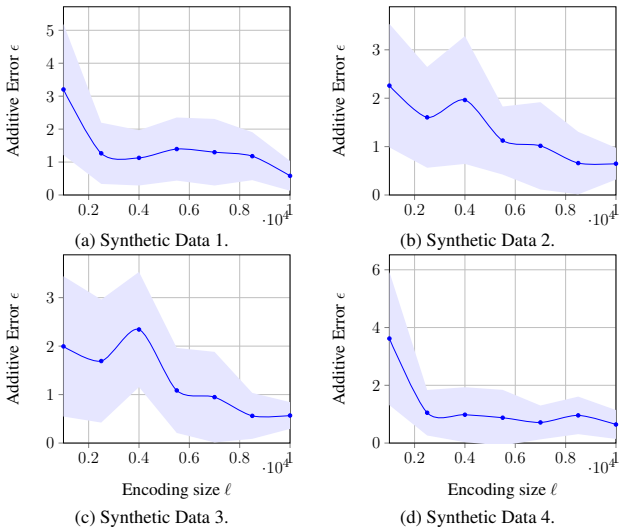


Figure 5: Additive error of FED-H w.r.t. encoding size ℓ.

(FDR) control (Foster & Stine, 2008) with FED-χ^2. In an online FDR control problem, a data analyst receives a stream of hypotheses on the database, or equivalently, a stream of p-values: $p_1, p_2, \cdots$. At each time t, the data analyst should pick a threshold α_t to reject the hypothesis when $p_t < \alpha_t$. The error metric is the false discovery rate, and the objective of online FDR control is to ensure that for any time t, the FDR up to time t is smaller than a pre-determined quantity. We use the SAFFRON procedure (Ramdas et al., 2018), the state-of-the-art online FDR control, for multiple hypothesis testing. The χ^2 results and corresponding p-values are calculated by FED-χ^2. We present the SAFFRON algorithm in Appendix C.

Each time, there are 100 independent hypotheses, with a probability of 0.5 that each hypothesis is either independent or correlated. The time sequence length is 100, and the number of clients is 10. The data are synthesized from a multivariate Gaussian distribution. For the correlated data, the covariance matrix is randomly sampled from a uniform distribution. For the independent data, the covariance matrix is diagonal, and its entries are randomly sampled from a uniform distribution.

At time t, we use FED-χ^2 to calculate the p-values p_t of all

the hypotheses, and then use the SAFFRON procedure to estimate the reject threshold α_t using p_t. The relationship between the average FDR and encoding size ℓ is shown in Fig. 4. We observe that the variance of independent runs is very small, so we omit the error bars. FED-χ^2 achieves good performance (FDR lower than 10%) when the encoding size l is larger than 200. In Fig. 4, we also provide the FDR result of the centralized χ^2-test as well as the true discovery rate (TDR, i.e., #correct reject / #should reject). In addition, we provide statistics for each encoding size l that was evaluated in Appendix M. The results indicate that by increasing the encoding size ℓ, FED-χ^2 can achieve comparable performance to the centralized χ^2-test. The results further demonstrate that FED-χ^2 can be employed in practice to facilitate online FDR control.

4.3. Performance of FED-H

We present the performance of FED-H on the four synthesized datasets in Fig. 5. We choose $m_x = 100000$ in this experiment, and other settings remain the same as Sec. 4.1. As expected, when the encoding size ℓ is properly large, the additive error of FED-H is small. For example, when $\ell = 10000$, the additive error is about 0.5, which is only about 4.5% of the ground truth. Note that we need larger ℓ in FED-H compared to FED-χ^2 to ensure that the multiplicative error in federated frequency moments estimation is small, which will in return give small additive error in FED-H. Nevertheless, FED-H still saves 90% of the communication than directly using secure aggregation. The results demonstrate the effectiveness and efficiency of FED-H in practice, and show that our framework can be naturally extended to compute other statistics that can be recast as frequency moments estimations in federated setting.

5. Conclusion & Discussion

This paper takes an important step towards designing nonlinear secure aggregation protocols in the federated setting. Specifically, we propose a universal secure protocol to evaluate frequency moments in the federated setting. We focus on important applications of the protocol: correlation test and entropy estimation. We give formal security proof and utility analysis and validate them empirically.

We also discuss potential future directions. FED-χ^2 and FED-H may leak the linear equation system of the marginal information and the encodings during server-client collusion, as discussed in Appendix H. And also we acknowledge inference attacks on aggregated encodings could be possible with prior knowledge of clients' data. However, quantifying information leakage in such cases remains challenging due to uncertainty about the required levels of prior information. Therefore, we deem it promising to enhance privacy guarantees for FED-χ^2 and FED-H by incorporating differential privacy techniques such as differentially private frequency moments estimation (Wang et al., 2021a) or adding calibrated discrete Gaussian noise (Canonne et al., 2020) to the users' local updates.

Acknowledgements

Qi Pang and Shuai Wang were supported in part by the HKUST 30 for 30 research initiative scheme under the the contract Z1283. Lun Wang and Dawn Song were partially supported by DARPA contract #N66001-15-C-4066, the Center for LongTerm Cybersecurity, and Berkeley Deep Drive. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the sponsors.

References

Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., and McMahan, B. cpsgd: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems*, 31, 2018.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948. PMLR, 2020.

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., and Raykova, M. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269, 2020.

Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643. PMLR, 2019.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Brunetti, M. and Renato, A. Old and new proofs of cramer's rule. *Applied Mathematical Sciences*, 8(133): 6689–6697, 2014.

Canonne, C. L., Kamath, G., and Steinke, T. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

Chen, W.-N., Choo, C. A. C., Kairouz, P., and Suresh, A. T. The fundamental price of secure aggregation in differen-

tially private federated learning. In *International Conference on Machine Learning*, pp. 3056–3089. PMLR, 2022.

Chen, Y., Luo, F., Li, T., Xiang, T., Liu, Z., and Li, J. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Information Sciences*, 522:69–79, 2020.

Choi, B., Sohn, J.-y., Han, D.-J., and Moon, J. Communication-computation efficient secure aggregation for federated learning. *arXiv preprint arXiv:2012.05433*, 2020a.

Choi, S. G., Dachman-Soled, D., Kulkarni, M., and Yerukhimovich, A. Differentially-private multi-party sketching for large-scale statistics. *Proceedings on Privacy Enhancing Technologies*, 2020(3):153–174, 2020b.

Cohen, J. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Dudoit, S., Shaffer, J. P., Boldrick, J. C., et al. Multiple hypothesis testing in microarray experiments. *Statistical Science*, 18(1):71–103, 2003.

Foster, D. P. and Stine, R. A. 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.

Govindaraj, Praveen. Credit Risk Classification Dataset: Is Customer Risky or Not Risky ? https://www.kaggle.com/praveengovi/credit-risk-classification-dataset. Online; accessed 22 April 2021.

Harrison, W. An entropy-based measure of software complexity. *IEEE Transactions on Software Engineering*, 18 (11):1025–1029, 1992.

Hayes, P. J. and Weinstein, S. P. Construe/tis: A system for content-based indexing of a database of news stories. In *IAAI*, volume 90, pp. 49–64, 1990.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

Indyk, P. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

Kairouz, P., Liu, Z., and Steinke, T. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pp. 5201–5212. PMLR, 2021.

Kassirer, J. P. Teaching clinical medicine by iterative hypothesis testing: let's preach what we practice, 1983.

Kim, J. H. and Ji, P. I. Significance testing in empirical finance: A critical review and assessment. *Journal of Empirical Finance*, 34:1–14, 2015.

Kohavi, R. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pp. 202–207, 1996.

Kohavi, Ronny and Becker, Barry. UCI Machine Learning Repository: Adult Data Set. https://archive.ics.uci.edu/ml/datasets/adult. Online; accessed 22 April 2021.

Kononenko, Igor and Cestnik, Bojan. UCI Machine Learning Repository: Lymphography Data Set. https://archive.ics.uci.edu/ml/datasets/Lymphography. Online; accessed 22 April 2021.

Ledoit, O. and Wolf, M. Robust performance hypothesis testing with the sharpe ratio. *Journal of Empirical Finance*, 15(5):850–859, 2008.

Lévy, P. and Lévy, P. *Calcul des probabilités*. Gauthier-Villars, 1925.

Li, P. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 10–19, 2008.

Lindell, Y. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pp. 277–346. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57048-8. doi: 10.1007/978-3-319-57048-8_6. URL https://doi.org/10.1007/978-3-319-57048-8_6.

Nyberg, K. Correlation theorems in cryptanalysis. *Discrete Applied Mathematics*, 111(1-2):177–188, 2001.

Nychis, G., Sekar, V., Andersen, D. G., Kim, H., and Zhang, H. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pp. 151–156, 2008.

Park, E., Ahn, J., and Yoo, S. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5456–5464, 2017.

Ramdas, A., Zrnic, T., Wainwright, M., and Jordan, M. Saffron: an adaptive algorithm for online control of the false discovery rate. In *International conference on machine learning*, pp. 4286–4294. PMLR, 2018.

Rosenberg, A. and Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 410–420, 2007.

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., and Arora, R. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020.

Sandeep Nandal. PyDroid. https://pypi.org/project/pydroid/, 2020. Online; accessed 24 April 2021.

Schlimmer, Jeff. UCI Machine Learning Repository: Mushroom Data Set. https://archive.ics.uci.edu/ml/datasets/Mushroom. Online; accessed 22 April 2021.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

So, J., Güler, B., and Avestimehr, A. S. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2021.

Spirtes, P., Glymour, C., Scheines, R., Kauffman, S., Aimale, V., and Wimberly, F. Constructing bayesian network models of gene expression networks from microarray data. 2000.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pp. 1–11, 2019.

Vempala, S. S. *The random projection method*, volume 65. American Mathematical Soc., 2005.

Wang, L., Jia, R., and Song, D. D2p-fed: Differentially private federated learning with efficient communication. *arxiv. org/pdf/2006.13039*, 2020.

Wang, L., Pinelis, I., and Song, D. Differentially private fractional frequency moments estimation with polylogarithmic space. *arXiv preprint arXiv:2105.12363*, 2021a.

Wang, L., Xu, S., Wang, X., and Zhu, Q. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10165–10173, 2021b.

Wikipedia. Contingency table — Wikipedia, the free encyclopedia, 2021. URL https://en.wikipedia.org/w/index.php?title=Contingency_table&oldid=1055319221. [Online; accessed 27-September-2022].

Wilson, R. B., Davis, D., and Mitchell, A. P. Rapid hypothesis testing with candida albicans through gene disruption with short homology regions. *Journal of bacteriology*, 181(6):1868–1874, 1999.

Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., and Ludwig, H. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pp. 13–23, 2019.

Yang, Y. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69–90, 1999.

Yang, Y. and Pedersen, J. O. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pp. 35. Nashville, TN, USA, 1997.

Zhang, J. and Yang, Y. Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 190–197, 2003.

Zhao, H., Lall, A., Ogihara, M., Spatscheck, O., Wang, J., and Xu, J. A data streaming algorithm for estimating entropies of od flows. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 279–290, 2007.

Zheng, Z., Wu, X., and Srihari, R. Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter*, 6(1):80–89, 2004.

Zhu, B., Wang, L., Pang, Q., Wang, S., Jiao, J., Song, D., and Jordan, M. I. Byzantine-robust federated learning with optimal statistical rates. In *International Conference on Artificial Intelligence and Statistics*, pp. 3151–3178. PMLR, 2023.

Appendix

A. Stable Distribution Refresher

A non-degenerate distribution is said to be stable if for X and Y sampled from the distribution, $aX + bY$ for some constants $a, b > 0$ has the same distribution up to location and scale parameters. Paul Léby first systematically study the stable distribution family in his master piece: Calcul des probabilités (Lévy & Lévy, 1925) so stable distribution is also referred to as Léby α-stable distribution. Stable distributions are parameterized by location μ, scale F, the stability parameter α and the skewness β. When $\alpha \neq 1$, the characteristic function is as below:

$$\phi_X(t) = \exp(it\mu - F|t|^\alpha(1 - \sqrt{-1}\beta \operatorname{sgn}(t) \tan(\frac{\pi\alpha}{2}))) \tag{12}$$

When $\alpha = 1$, the characteristic function is given by:

$$\phi_X(t) = \exp(it\mu - F|t|^\alpha(1 + \frac{2}{\pi}\sqrt{-1}\beta \operatorname{sgn}(t) \log|t|)) \tag{13}$$

In the main text, we only consider a subset of stable distributions where $\mu = 0$ and $\alpha \neq 1$.

Stable distribution family contains many familiar distributions. For example, 1-stable distribution is Cauchy distribution, 2-stable distribution is Gaussian distribution, and 1/2-stable distribution is known as Lev́y distribution.

Stable distributions also have discrete analogues defined by their probability generating function

$$G(t) = exp(-Ft^\alpha), \tag{14}$$

where F is the scale and α is the stability parameter. However, for discrete stable distribution, the support domain of α is $(0, 1]$ instead of $(0, 2]$.

B. Secure Aggregation Refresher

The secure aggregation protocol from (Bell et al., 2020) is presented in Alg. 4. The first step of the protocol is to generate a k-regular graph G, where the n vertices are the clients participating in the protocol. The server runs a randomized graph generation algorithm InitSecureAgg presented in Alg. 3 that takes the number of clients n and samples output (G, t, k) from a distribution $\mathcal{D}$. In Alg. 3, we uniformly rename the nodes of a graph known as a Harary graph defined in Definition 4 with n nodes and k degrees. The graph G is constructed by sampling k neighbours uniformly and without replacement from the set of remaining $n - 1$ clients. We choose $k = \mathcal{O}(log(n))$, which is large enough to hide the updates inside the masks. t is the threshold of the Shamir's Secret Sharing.

In the second step, the edges of the graph determine pairs of clients, each of which runs key agreement protocols to share random keys. The random keys will be used by each party to derive a mask for her input and enable dropouts.

In the third step, each client $c_i, i \in A_1$ sends secret share to its neighbors. In the fourth step, the server checks whether the clients dropout exceeds the threshold δ, and lets the clients know their neighbors who didn't dropout.

In the fifth step, each pair (i, j) of connected clients in G runs a λ-secure key agreement protocol $s_{i,j} = \mathcal{KA}.Agree(sk_i^1, pk_j^1)$ which uses the key exchange in the previous step to derive a shared random key $s_{i,j}$. The pairwise masks $\mathbf{m}_{i,j} = F(s_{i,j})$ can be computed, where F is the pseudorandom generator (PRG). If the semi-honest server announces dropouts and later some masked inputs of the claimed dropouts arrive, the server can recover the inputs. To prevent this happening, another level of masks, called self masks, $\mathbf{r}_i$ is added to the input. Thus, the input of client c_i is: $\mathbf{y}_i = \mathbf{e}_i + \mathbf{r}_i - \sum_{j \in N_G(i), j<i} \mathbf{m}_{i,j} + \sum_{j \in N_G(i), j>i} \mathbf{m}_{i,j}$.

Steps 6–8 deal with the clients dropout by recovering the self masks $\mathbf{r}_i$ of clients who are still active and pairwise masks $\mathbf{m}_{i,j}$ of the clients who have dropped out. Finally, the server can cancel out the pairwise masks and subtract the self masks in the final sum: $\sum_{i \in A_2'}(\mathbf{y}_i - \mathbf{r}_i + \sum_{j \in NG(i) \cap (A_1' \setminus A_2'), 0<j<i} \mathbf{m}_{i,j} - \sum_{j \in NG(i) \cap (A_1' \setminus A_2'), i<j \leq n} \mathbf{m}_{i,j})$.

Definition 4 (Harary(n, k) Graph)**.** *Let* Harary(n, k) *denotes a graph with n nodes and degree k. This graph has vertices $V = [n]$ and an edge between two distinct vertices i and j if and only if $j - i \pmod n \leq (k + 1)/2$ or $j - i \pmod n \geq n - k/2$.*

C. SAFFRON Procedure Refresher

In Sec. 4.2, we adopt the SAFFRON procedure (Ramdas et al., 2018) to perform online FDR control. SAFFRON procedure is currently the state of the arts for multiple hypothesis testing. In Alg. 5, we formally present the SAFFRON algorithm.

The initial error budget for SAFFRON is $(1 - \lambda_1 W_0) < (1 - \lambda_1 \alpha)$, and this will be allocated to different tests over time. The sequence $\{\lambda_j\}_{j=1}^{\infty}$ is defined by g_t and λ_j serves as a weak estimation of α_j. g_t can be any coordinate wise non-decreasing function (line 8 in Alg. 5). $R_j := I(p_j < \alpha_j)$ is the indicator for rejection, while $C_j := I(p_j < \lambda_j)$ is the indicator for candidacy. τ_j is the jth rejection time. For each p_t, if $p_t < \lambda_t$, SAFFRON adds it to the candidate set C_t and sets the candidates after the jth rejection (lines 9-10 in Alg. 5). Further, the α_t is updated by several parameters like current wealth, current total rejection numbers, the current size of the candidate set, and so on (lines 11-14 in Alg. 5). Then, the decision R_t is made according to the updated α_t (line 15 in Alg. 5).

The hyper-parameters for the SAFFRON procedure in online false discovery rate control of Sec. 4 are aligned with the setting in (Ramdas et al., 2018). The target FDR level is $\alpha = 0.05$, the initial wealth is $W_0 = 0.0125$, and γ_j is calculated in the following way: $\gamma_j = \frac{1/(j+1)^{1.6}}{\sum_{j=0}^{10000} 1/(j+1)^{1.6}}$.

D. Proof for Theorem 1

Proof for Theorem 1. To prove Theorem 1, we need the following lemma.

Lemma 1 (Security of secure aggregation protocol)**.** *Let* SECUREAGG *be the secure aggregation protocol in Alg. 4 of*

Algorithm 3 INITSECUREAGG: Generate Initial Graph for SECUREAGG.

Function INITSECUREAGG(n)**:**
- ▷ n: Number of nodes.
- ▷ t: Threshold of Shamir's Secret Sharing.
- $k = \mathcal{O}(log(n))$.
- Let $H =$ HARARY(n, k).
- Sample a random permutation $\pi : [n] \to [n]$.
- Let G be the set of edges $\{(\pi(i), \pi(j)) | (i, j) \in H\}$.
- **return** (G, t, k)

Algorithm 5 SAFFRON Procedure.

Function SAFFRONPROCEDURE$(\{p_1, p_2, \cdots\}, \alpha, W_0, \{\gamma_j\}_{j=0}^{\infty})$**:**
- ▷ $\{p_1, p_2, \cdots\}$: Stream of p-values computed by FED-χ^2.
- ▷ α: Target FDR level.
- ▷ W_0: Initial wealth.
- ▷ $\{\gamma_j\}_{j=0}^{\infty}$: Positive non-increasing sequence summing to one.
- $i \leftarrow 0$ `// Set rejection number.`
- **for** *each p-value* $p_t \in \{p_1, p_2, \cdots\}$ **do**
  - $\lambda_t \leftarrow g_t(R_{1:t-1}, C_{1:t-1})$
  - $C_t \leftarrow I(p_t < \lambda_t)$ `// Set the indicator for candidacy` C_t.
  - $C_{j+} \leftarrow \sum_{i=\tau_j+1}^{t-1} C_i$ `// Set the candidates after the` j^{th} `rejection.`
  - **if** $t = 1$ **then**
    - $\alpha_1 \leftarrow (1 - \lambda_1)\gamma_1 W_0$
  - **else**
    - $\alpha_t \leftarrow (1 - \lambda_t)(W_0 \gamma_{t-C_{0+}} + (\alpha - W_0)\gamma_{t-\tau_1-C_{1+}} + \sum_{j \geq 2} \alpha \gamma_{t-\tau_j-C_{j+}})$
  - $R_t \leftarrow I(p_t \leq \alpha_t)$ `// Output` R_t.
  - **if** $R_t = 1$ **then**
    - $i \leftarrow i + 1$ `// Update rejection number.`
    - $\tau_i \leftarrow t$ `// Set the` i^{th} `rejection time.`
- **return** $\{R_0, R_1, \cdots\}$

Algorithm 4 SECUREAGG: Secure Aggregation Protocol. (Algorithm 2 from Bell et al. (2020))

Function SECUREAGG $(\{\boldsymbol{e}_i\}_{i\in[n]})$ **:**

▷ Parties: Clients $c_1, \cdots, c_n$, and Server.
▷ l: Vector length.
▷ $\mathbb{X}^l$: Input domain, $\mathbf{e}_i \in \mathbb{X}^l$.
▷ $F : \{0,1\}^\lambda \to \mathbb{X}^l$: PRG.
▷ *We denote by A_1, A_2, A_3 the sets of clients that reach certain points without dropping out. Specifically A_1 consists of the clients who finish step (3), A_2 those who finish step (5), and A_3 those who finish step (7). For each A_i, A'_i is the set of clients for which the server sees they have completed that step on time.*

(1) The server runs $(G, t, k) =$ INITSECUREAGG (n), where G is a regular degree-k undirected graph with n nodes. By $N_G(i)$ we denote the set of k nodes adjacent to c_i (its neighbors).
(2) Client $c_i, i \in [n]$, generates key pairs (sk_i^1, pk_i^1), (sk_i^2, pk_i^2) and sends (pk_i^1, pk_i^2) to the server who forwards the message to $N_G(i)$.
(3) **for** *each Client* $c_i, i \in A_1$ **do**

- Generates a random PRG seed b_i.
- Computes two sets of shares:

$$H_i^b = \{h_{i,1}^b, \cdots, h_{i,k}^b\} = ShamirSS(t, k, b_i)$$

$$H_i^s = \{h_{i,1}^s, \cdots, h_{i,k}^s\} = ShamirSS(t, k, sk_i^1)$$

- Sends to the server a message $m = (j, c_{i,j})$, where $c_{i,j} = \mathcal{E}_{auth}.Enc(k_{i,j}, (i||j||h_{i,j}^b||h_{i,j}^s))$, and $k_{i,j} = \mathcal{KA}.Agree(sk_i^2, pk_j^2)$, for each $j \in N_G(i)$.

(4) The server aborts if $|A'_1| < (1-\delta)n$ and otherwise forwards $(j, c_{i,j})$ to client c_j who deduces $A'_1 \cap N_G(j)$.
(5) **for** *each Client* $c_i, i \in A_2$ **do**

- Computes a shared random PRG seed $s_{i,j}$ as $s_{i,j} = \mathcal{KA}.Agree(sk_i^1, pk_j^1)$.
- Computes masks $\mathbf{m}_{i,j} = F(s_{i,j})$ and $\mathbf{r}_i = F(b_i)$.
- Sends to the server their masked input

$$\mathbf{y}_i = \mathbf{e}_i + \mathbf{r}_i - \sum_{j\in[n], j<i} \mathbf{m}_{i,j} + \sum_{j\in[n], j>i} \mathbf{m}_{i,j}$$

(6) The server collects masked inputs. It aborts if $|A'_2| < (1-\delta)n$ and otherwise sends $(A'_2 \cup N_G(i), (A_1 \backslash A'_2) \cup N_G(i))$ to every client $c_i, i \in A'_2$.
(7) Client $c_j, j \in A_3$ receives (R_1, R_2) from the server and sends $\{(i, h_{i,j}^b)\}_{i\in R_1} \cup \{(i, h_{i,j}^s)\}_{i\in R_2}$ obtained by decrypting the $c_{i,j}$ received in Step (3).
(8) The server aborts if $|A'_3| < (1-\delta)n$ and otherwise:

- Collects, for each client $c_i, i \in A'_2$, the set B_i of all shares in H_i^b sent by clients in A_3. Then aborts if $|B_i| < t$ and otherwise recovers b_i and $\mathbf{r}_i$ using the t shares received which came from the lowest client IDs.
- Collects, for each client $c_i, i \in (A_1 \backslash A'_2)$, the set S_i of all shares in H_i^s sent by clients in A_3. Then aborts if $|S_i| < t$ and otherwise recovers sk_i^1 and $\mathbf{m}_{i,j}$.
- **return** $\sum_{i\in A'_2}(\mathbf{y}_i - \mathbf{r}_i + \sum_{j\in NG(i)\cap(A'_1\backslash A'_2), 0<j<i} \mathbf{m}_{i,j} - \sum_{j\in NG(i)\cap(A'_1\backslash A'_2), i<j\le n} \mathbf{m}_{i,j})$.

Appendix B instantiated with cryprographic security parameter λ. There exists a probabilistic polynomial-time (PPT) simulator SIMSA *such that for all clients $\mathcal{C}$, the number of clients n, and the aggregated encoding $\boldsymbol{e}$, the output of* SIMSA *is perfectly indistinguishable from the view of the real server,* i.e., $\text{SECUREAGG}_{\mathcal{C}} \approx_\lambda \text{SIMSA}(\boldsymbol{e}, n)$.

Lemma 1 is derived from the security analysis of our employed secure aggregation protocol (Theorem 3.6 in Bell et al. (2020)), which establishes that the secure aggregation protocol securely conceals the individual information in the aggregated result. With this lemma, we are able to prove the theorem for federated correlation test by presenting a sequence of hybrids that begin with real protocol execution and end with simulated protocol execution. We demonstrate that every two consecutive hybrids are indistinguishable, illustrating that the hybrids are indistinguishable according to transitivity.

- HYB_1 This is the view of the server in the real protocol execution, $\text{REAL}_{\mathcal{C}}$.
- HYB_2 In this hybrid, we replace the view during the execution of each $\text{SECUREAGG}(\{\mathbf{v}_x^{(i)}\}_{i\in[n]})$ in line 3 of Alg. 2 with the output of $\text{SIMSA}(\mathbf{v}_x, n)$ one by one correspondingly. According to Lemma 1, each replacement does not change the indistinguishability. Hence, HYB_2 is indistinguishable from HYB_1.
- HYB_3 Similar to HYB_2, we replace the view during the execution of each $\text{SECUREAGG}(\{\mathbf{v}_y^{(i)}\}_{i\in[n]})$ in line 4 of Alg. 2 with the output of $\text{SIMSA}(\mathbf{v}_y, n)$ one by one. According to Lemma 1, HYB_3 is indistinguishable from HYB_2.
- HYB_4 In this hybrid, we replace the view during the execution of $\text{SECUREAGG}(\{\mathbf{e}_i\}_{i\in[n]})$ in line 15 of Alg. 2 with the $\text{SIMSA}(\mathbf{e}, n)$. This hybrid is the output of SIM. According to Lemma 1, HYB_4 is indistinguishable from HYB_3. □

E. Proof for Utility

Proof for Theorem 2. First, we introduce the following lemma from Li (2008).

Lemma 2 (Tail bounds of geometric mean estimator (Li, 2008))**.** *The right tail bound of geometric mean estimator is:*

$$\mathbb{P}(\hat{s}_{\chi^2} - s_{\chi^2} > \epsilon s_{\chi^2}) \leq \exp(-\ell \eta^R_{\alpha,\epsilon}), \tag{15}$$

where $\eta^R_{\alpha,\epsilon} = C_1 \log(1+\epsilon) - C_1\gamma_e(\alpha-1) - \log(\frac{2}{\pi}\Gamma(\alpha C_1)\Gamma(1-C_1)\sin(\frac{\pi\alpha C_1}{2}))$*, α is a constant in our setting,* $C_1 = \frac{2}{\pi}\tan^{-1}(\frac{\log(1+\epsilon)}{(2+\alpha^2)\pi/6})$*, and γ_e is the Euler's constant.*

The left tail bound of the geometric mean estimator is:

$$\mathbb{P}(\hat{s}_{\chi^2} - s_{\chi^2} < -\epsilon s_{\chi^2}) \leq \exp(-\ell \eta^L_{\alpha,\epsilon}), \tag{16}$$

where $\ell > \ell_0$*,* $\eta^L_{\alpha,\epsilon} = -C_2\log(1-\epsilon) - \log(-\frac{2}{\pi}\Gamma(-\alpha C_2)\Gamma(1+C_2)\sin(\frac{\pi\alpha C_2}{2})) - \ell_0 C_2 \log(\frac{2}{\pi}\Gamma(\frac{\alpha}{\ell_0})\Gamma(1-\frac{1}{\ell_0})\sin(\frac{\pi}{2}\frac{\alpha}{\ell_0}))$*, and* $C_2 = \frac{12}{\pi^2}\frac{\epsilon}{(2+\alpha^2)}$.

With Lemma 2, taking $\eta_{\alpha,\epsilon} \leq \min(\eta^L_{\alpha,\epsilon}, \eta^R_{\alpha,\epsilon})$ and $\delta = 2\exp(-\ell\eta_{\alpha,\epsilon})$, we are able to prove $\mathbb{P}[\hat{s}_{\chi^2} < (1-\epsilon)s_{\chi^2} \vee \hat{s}_{\chi^2} > (1+\epsilon)s_{\chi^2}] \leq \delta$ with union bound, which is achieved when $\ell = \frac{1}{\eta_{\alpha,\epsilon}}\log(2/\delta)$. □

F. Proof for Communication & Computation Cost

In this section, we prove Theorem 3 and Theorem 4.

Theorem 3 (Communication Cost). Let Π be an instantiation of Alg. 2 with secure aggregation protocol from (Bell et al., 2020), then (1) the client-side communication cost is $\mathcal{O}(\log n + m_x + m_y + \ell)$; (2) the server-side communication cost $\mathcal{O}(n\log n + nm_x + nm_y + n\ell)$.

Proof sketch for Theorem 3. Each client performs k key agreements ($\mathcal{O}(k)$ messages, line 9 in Alg. 4) and sends 3 masked inputs ($\mathcal{O}(m_x + m_y + \ell)$ complexity, lines 3, 4, 15 in Alg. 2 and line 10 in Alg. 4). Thus, the client communication cost is $\mathcal{O}(\log n + m_x + m_y + \ell)$.

The server receives or sends $\mathcal{O}(\log n + m_x + m_y + \ell)$ messages to each client, so the server communication cost is $\mathcal{O}(n\log n + nm_x + nm_y + n\ell)$. □

Theorem 4 (Computation Cost). Let Π be an instantiation of Alg. 2 with secure aggregation protocol from (Bell et al., 2020), then (1) the client-side computation cost is $\mathcal{O}(m_x\log n + m_y\log n + \ell\log n + m\ell)$; (2) the server-side computation cost is $\mathcal{O}(m_x + m_y + \ell)$.

Proof sketch for Theorem 4. Each client computation can be broken up as k key agreements ($\mathcal{O}(k)$ complexity, line 9 in Alg. 4), generating masks $\mathbf{m}_{i,j}$ for all neighbors c_j ($\mathcal{O}(k(m_x + m_y + \ell))$ complexity, lines 3, 4, 15 in Alg. 2 and line 10 in Alg. 4), sampling encoding matrix $\mathbf{P}$ cost $\mathcal{O}(m\ell)$, line 13 in Alg. 2, and encoding computation cost $\mathcal{O}(m\ell)$ (line 14 in Alg. 2). Thus, the client computation cost is $\mathcal{O}(m_x \log n + m_y \log n + \ell \log n + m\ell)$.

The server-side follows directly from the semi-honest computation analysis in Bell et al. (2020). The extra $\mathcal{O}(\ell)$ term is the complexity of the geometric mean estimator.

□

G. Proof for Utility of FED-H

In this section, we prove the utility of FED-H presented in Eq. 11.

Proof for Eq. 11. Given a proper set of ϵ, ℓ, and Δ, we have $\widehat{\sum_{x\in\mathcal{X}} v_x^{1+\Delta}} \in (1 \pm \epsilon) \sum_{x\in\mathcal{X}} v_x^{1+\Delta}$ and $\widehat{\sum_{x\in\mathcal{X}} v_x^{1-\Delta}} \in (1 \pm \epsilon) \sum_{x\in\mathcal{X}} v_x^{1-\Delta}$.

Thus: $\hat{H} = -\frac{1}{2\Delta}(\widehat{\sum_{x\in\mathcal{X}} v_x^{1+\Delta}} - \widehat{\sum_{x\in\mathcal{X}} v_x^{1-\Delta}}) \in \tilde{H} \pm \frac{(m_x^{\Delta}+1)}{2\Delta}\epsilon$, where $\tilde{H}$ is the approximated entropy using finite difference approximation.

Now we calculate the bound of $|\tilde{H} - H|$.

Consider $f(t) = v_x^t$, then $f'(t) = ln(v_x)v_x^t$, $f'(1) = ln(v_x)v_x$ is the result we are interested in.

So we estimate $f'(t)$ near $t = 1$ using finite difference approximations, and the error is derived as follows:

According to Taylor series: $f(t+h) = f(t) + hf'(t) + \frac{h^2}{2}f''(t) + \frac{h^3}{6}f'''(t+h_1)$, where $h_1 \in [0, h]$, $h > 0$.

$f(t-h) = f(t) - hf'(t) + \frac{h^2}{2}f''(t) - \frac{h^3}{6}f'''(t-h_2)$, where $h_2 \in [0, h]$, $h > 0$.

Taking $t = 1$, $h = \Delta$, and we subtract the above equations: $f(1+\Delta) - f(1-\Delta) = 2\Delta f'(1) + \frac{\Delta^3}{6}f'''(1+h_0)$, $h_0 \in [-\Delta, \Delta]$.

Thus, the additive error of the finite difference approximation is: $\epsilon_{FD} = |\frac{f(1+\Delta)-f(1-\Delta)}{2\Delta} - f'(1)| = |\frac{\Delta^2}{6}f'''(1+h_0)|$.

Since $f'''(t) = ln^3(v_x)v_x^t$, $\epsilon_{FD} = -\frac{\Delta^2}{6}ln^3(v_x)v_x^{1+h_0}$. Given that $v_x \in (0, 1)$, $-ln^3(v_x) > 0$, and $h_0 \in [-\Delta, \Delta]$, we have: $\epsilon_{FD} \leq -\frac{\Delta^2}{6}ln^3(v_x)v_x^{1-\Delta}$. Given that $v_x \in (0, 1)$, we have $\epsilon_{FD} \leq \frac{9\Delta^2}{2(1-\Delta)^3e^3}$. So the additive error of the finite difference approximation in entropy estimation is bounded, and we do not have additional assumptions on the frequency v_x.

Thus, $|\hat{H} - H| \in H \pm (\frac{(m_x^{\Delta}+1)}{2\Delta}\epsilon + \frac{9\Delta^2}{2(1-\Delta)^3e^3})$.

□

H. Further Security Analysis when Collusion Happens

We have shown that Alg. 2 provides strong security guarantee when there is no collusion between the clients and the server. That is, the server only knows the non-private marginal distribution of the contingency table and the final aggregated results. In the following section, we will analyze the leakage of Alg. 2 when the collusion happens to demonstrate that FED-χ^2 provides strong privacy guarantee and also help the readers better understand our protocol.

Remark: what does Alg. 2 leak when collusion between the server and the client happens? If the server colludes with one client, then it knows the random seed r (line 12 of Alg. 2) used to generate the projection matrix $\mathbf{P}$. In the following, we will analyze the leakage of client private data when the server knows $\mathbf{P}$.

By Theorem 1, we show that individual updates of clients are perfectly hidden in the aggregated results and FED-χ^2 leaks

no more than a linear equation system if the server knows $\mathbf{P}$:

$$\begin{cases} \mathbf{P} \times \mathbf{v} & = \mathbf{e}^T \\ \mathbf{J}_{1,m_y} \times \mathbf{V}^T & = \mathbf{v}_x^T \\ \mathbf{J}_{1,m_x} \times \mathbf{V} & = \mathbf{v}_y^T \end{cases}, \tag{17}$$

where $\mathbf{J}_{1,m_x}$ and $\mathbf{J}_{1,m_y}$ are $1 \times m_x$ and $1 \times m_y$ unit matrices, $\mathbf{V}$ is an $m_x \times m_y$ matrix whose elements are $\{v_{xy}\}$, and $\mathbf{v}$ is the flattened vector of $\mathbf{V}$.

To understand (17), $\mathbf{v}$ (or $\mathbf{V}$) is sensitive and all the other matrices and vectors are already known to the server. Also note that due to the requirement of secure aggregation, all the values in (17) are discretized into a finite field. Thus, the server can solve the system of equations (17) on a finite field to get information about $\mathbf{v}$. The following theorem establishes an important fact: the above equation system has a large solution space, which conceals the real joint distribution.

Proposition 1. *Given a projection matrix $\boldsymbol{P} \in \mathbb{Z}_q^{\ell \times m}$, $\boldsymbol{v}_x \in \mathbb{Z}_q^{m_x}$, $\boldsymbol{v}_y \in \mathbb{Z}_q^{m_y}$ and $\boldsymbol{e} \in \mathbb{Z}_q^{\ell}$, if $m > \ell + m_x + m_y$, there are at least $q^{m-\ell-m_x-m_y}$ solutions to the system of equations (17).*

Proof sketch for Proposition 1. The system of linear equations on $\mathbb{Z}_q$ contains $m_x + m_y + \ell$ equations and m variables. Given $m > m_x + m_y + \ell$, the rank of the coefficient matrix is no more than $m_x + m_y + \ell$. According to the Rouché–Capelli theorem (Brunetti & Renato, 2014) on finite fields, the solution forms a at least $m - m_x - m_y - \ell$-dimensional traslation of subspace of $\mathbb{Z}_q^m$. As a result, we know that the solution space contains at least $q^{m-\ell-m_x-m_y}$ solution vectors. □

Theorem 1 shows an important fact that the joint distribution is hidden in a subspace with exponential possible distributions. Although the collusion between the client and the server is not likely to happen in the cross-silo federated settings (consider our example in Sec. 3.1) and thus not considered in our threat model, we still show that Alg. 2 practically enforce privacy given the considerably large size of the solution space.

I. Details of Datasets

The details for the real-world datasets used in Sec. 4.1 are provided in Table 1. The license of Credit Risk Classification (Govindaraj, Praveen) is CC BY-SA 4.0, the license of German Traffic Sign (Houben et al., 2013) is CC0: Public Domain. Other datasets without a license are from UCI Machine Learning Repository (Dua & Graff, 2017).

Table 1: Dataset details.

ID	Data	Attr #1	A#1 Cat	Attr #2	A#2 Cat
1	Adult Income (Kohavi, 1996; Kohavi, Ronny and Becker, Barry)	Occupation	14	Native Country	41
2	Credit Risk Classification (Govindaraj, Praveen)	Feature 6	14	Feature 7	11
3	Credit Risk Classification (Govindaraj, Praveen)	Credit Product Type	28	Overdue Type I	35
4	Credit Risk Classification (Govindaraj, Praveen)	Credit Product Type	28	Overdue Type II	35
5	Credit Risk Classification (Govindaraj, Praveen)	Credit Product Type	28	Overdue Type III	36
6	German Traffic Sign (Houben et al., 2013)	Image Width	219	Traffic Sign	43
7	German Traffic Sign (Houben et al., 2013)	Image Height	201	Traffic Sign	43
8	German Traffic Sign (Houben et al., 2013)	Upper left X coordinate	21	Traffic Sign	43
9	German Traffic Sign (Houben et al., 2013)	Upper left Y coordinate	16	Traffic Sign	43
10	German Traffic Sign (Houben et al., 2013)	Lower right X coordinate	204	Traffic Sign	43
11	German Traffic Sign (Houben et al., 2013)	Lower right Y coordinate	186	Traffic Sign	43
12	Mushroom (Schlimmer, Jeff)	Cap color	10	Odor	9
13	Mushroom (Schlimmer, Jeff)	Gill color	12	Stalk color above ring	9
14	Mushroom (Schlimmer, Jeff)	Stalk color below ring	9	Ring Type	8
15	Mushroom (Schlimmer, Jeff)	Spore print color	9	Habitat	7
16	Lymphography (Kononenko, Igor and Cestnik, Bojan)	Structure Change	8	No. of nodes	8

J. Full Version of Fig. 1

In this section, we present the full version of Fig. 1 on all the 20 datasets we evaluate in Fig. 6.

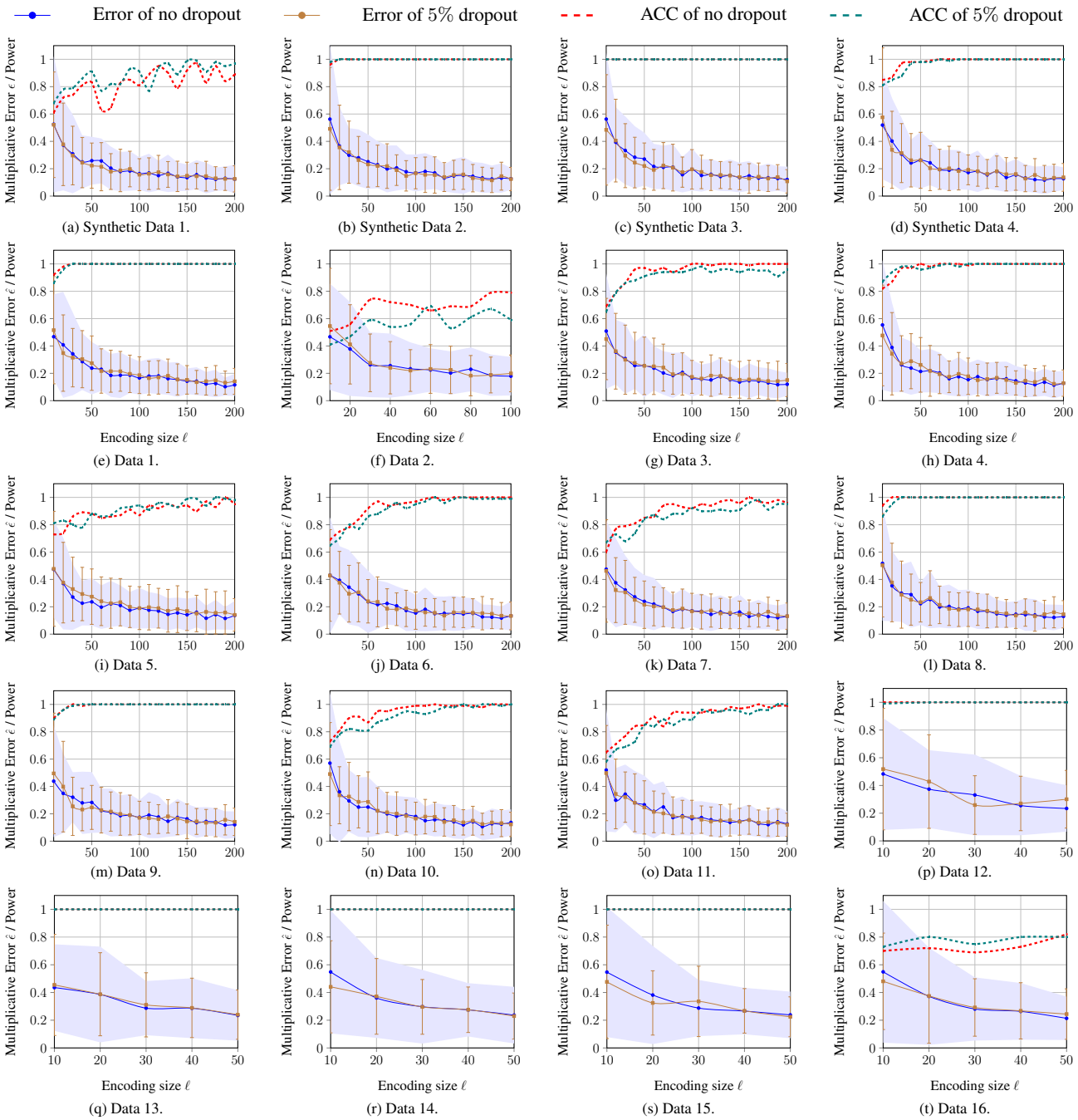


Figure 6: Multiplicative error and power of FED-χ^2 w.r.t. encoding size ℓ with and without dropout.

K. Details of Regression Models

The details of the regression models trained in feature selection in Sec. 4.2 is reported in Table 2. The training and testing splits are the same for FED-χ^2, centralized χ^2-test and model without feature selection (i.e. there are 17,262 training and 4,316 test documents). We use the same learning rate; random seed and all other settings are also the same to make the comparison fair. We get the result of Fig. 3 and the models are all trained on NVIDIA GeForce RTX 3090.

Table 2: Model details.

Task	Model Size	Learning Rate	Random Seed
FED-χ^2	40000×20	0.1	0
Centralized χ^2-test	40000×20	0.1	0
Without Feature Selection	167135×20	0.1	0

L. Further Results on FED-χ^2 with Dropouts

We present the results of 10%, 15%, and 20% clients dropout in Fig. 7. The results further show that FED-χ^2 can tolerate a considerable portion of clients dropout in Round 2 of Alg. 2.

M. Further Results for Online FDR Control

Table 3: Detailed results for online FDR control.

	#should reject	#should accept	#correct reject	#false reject
FED-χ^2, $l = 10$	5,900	4,100	5,144	1,392
FED-χ^2, $l = 25$	5,900	4,100	5,328	1,356
FED-χ^2, $l = 50$	5,900	4,100	5,325	1,143
FED-χ^2, $l = 100$	5,900	4,100	5,398	943
FED-χ^2, $l = 150$	5,900	4,100	5,393	765
FED-χ^2, $l = 200$	5,900	4,100	5,377	687
FED-χ^2, $l = 250$	5,900	4,100	5,328	615
FED-χ^2, $l = 300$	5,900	4,100	5,361	556
centralized χ^2-test	5,900	4,100	5,294	408

In this section, we provide further results for online FDR control. As we have shown in Fig. 4, FED-χ^2 achieves good performance when the encoding size l is larger than 200. In addition, we provide statistics for each encoding size l that was evaluated in Table 3. These results demonstrate that FED-χ^2 performs well and is comparable to the centralized χ^2-test when the encoding size l is increased.

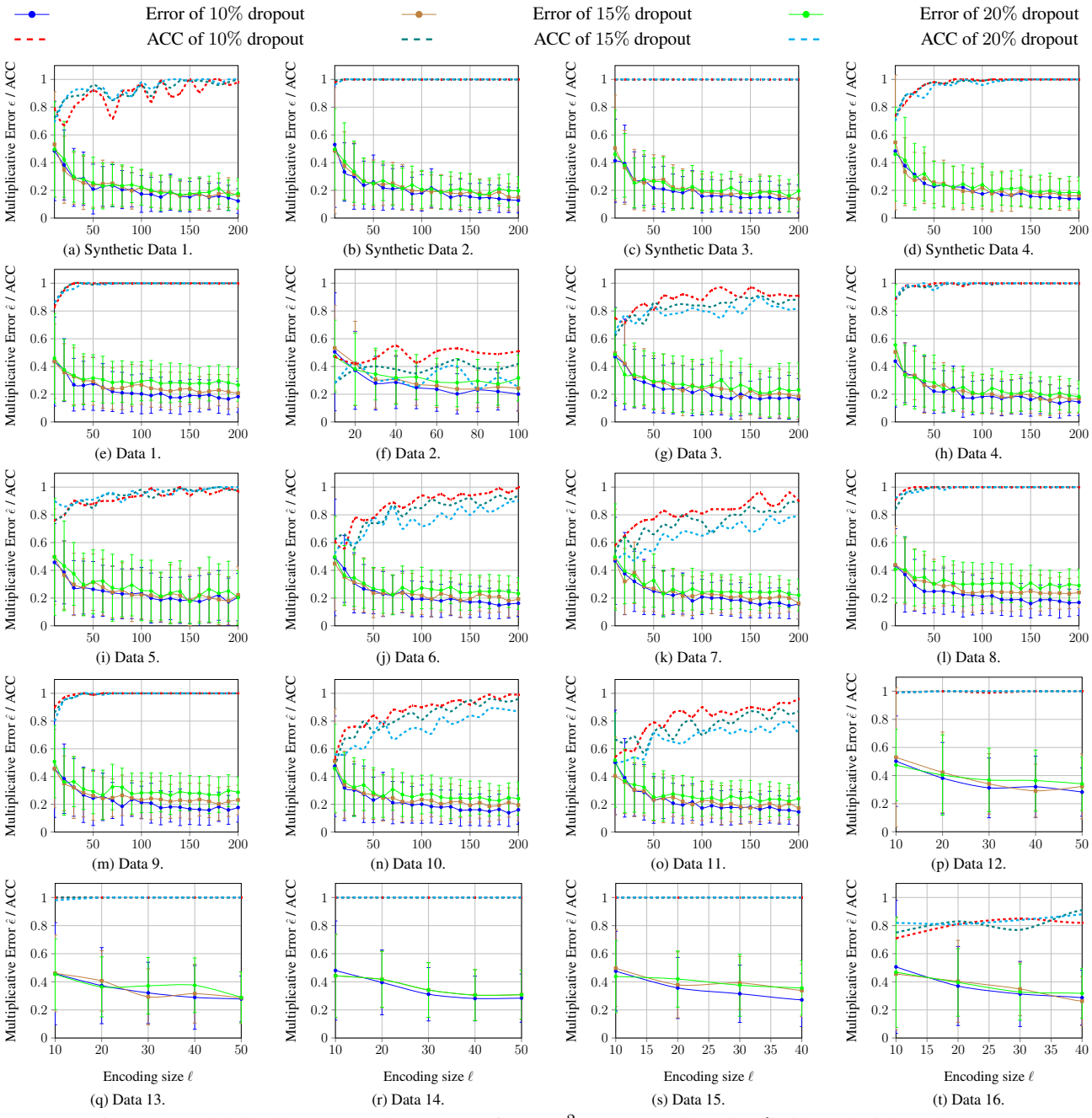
N. Further Results for Feature Selection

Our results in Sec. 4.2, paragraph **Feature Selection**, demonstrate that FED-χ^2 performs well when encoding size $l = 50$. We conduct experiments with different encoding sizes l to further assess their effect on FED-χ^2's performance. In Fig. 8, we present the effect of encoding size l on the ratio of the commonly-selected features between the original centralized χ^2-test and FED-χ^2. A larger ratio of commonly-selected features means that FED-χ^2 performs more closely to the original centralized χ^2-test. And if the ratio is 1, these two algorithms select the identical features. The results in Fig. 8 show that when the encoding size l increases, the performance of FED-χ^2 approaches that of the original centralized χ^2-test.

Similar to Sec. 4.2, we evaluate FED-χ^2's performance under different encoding sizes l by training the model with the features selected by FED-χ^2. Fig. 9 shows the results. When trained with FED-χ^2-selected features, the model can achieve comparable accuracy to the model trained with features selected by the original centralized χ^2-test. Also, consistent with the results in Fig. 3 in Sec. 4.2, we see that when the encoding size $l \geq 25$, models trained by FED-χ^2-selected features achieve higher accuracy than that of the models without feature selection. These results further demonstrate the effectiveness of FED-χ^2.

O. Influence of Finite Field Size

As shown in Fig. 10, we test the performance of FED-χ^2 under different finite field size q. We observe that when $q \in \{2^{16}, 2^{32}, 2^{64}\}$, there is almost no difference in the performance. The result shows that FED-χ^2 is numerically stable.



(a) Synthetic Data 1. (b) Synthetic Data 2. (c) Synthetic Data 3. (d) Synthetic Data 4.

(e) Data 1. (f) Data 2. (g) Data 3. (h) Data 4.

(i) Data 5. (j) Data 6. (k) Data 7. (l) Data 8.

(m) Data 9. (n) Data 10. (o) Data 11. (p) Data 12.

(q) Data 13. (r) Data 14. (s) Data 15. (t) Data 16.

Figure 7: Multiplicative error and accuracy of FED-χ^2 w.r.t. encoding size ℓ with and without dropout.

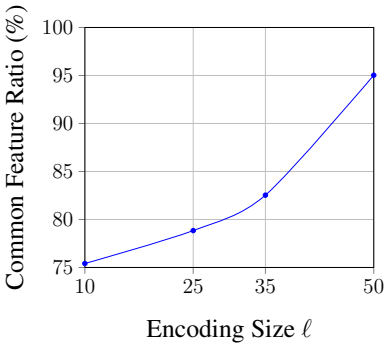


Figure 8: Ratio of commonly-selected features between FED-χ^2 and original centralized χ^2-test.

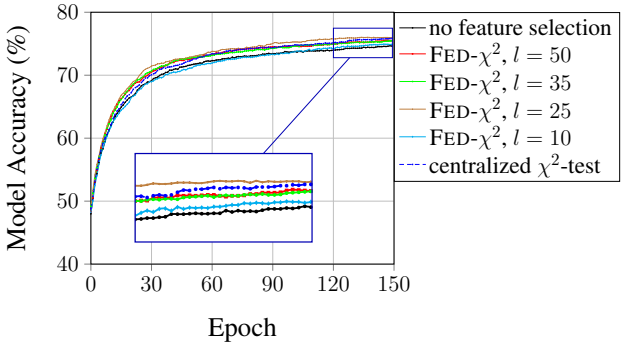


Figure 9: Accuracy of model trained w/ FED-χ^2-select features under different encoding size l.

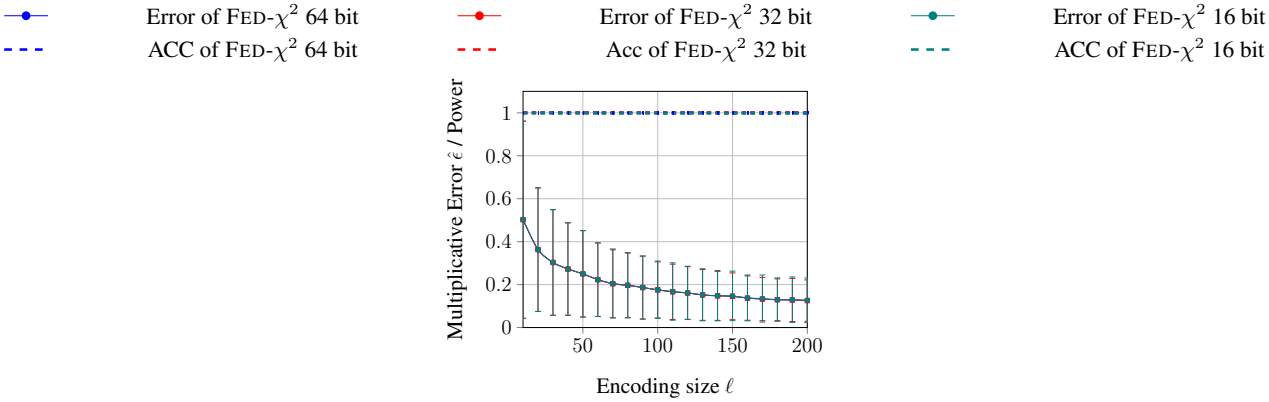


Figure 10: Performance of FED-χ^2 with different finite field size on synthetic data 3.