Secure Federated Feature Selection

Lun Wang[1*], **Qi Pang**[2*], **Shuai Wang**[2], **Dawn Song**[1]

[1] UC Berkeley, [2] HKUST

wanglun@berkeley.edu, qpangaa@cse.ust.hk, shuaiw@cse.ust.hk, dawnsong@cs.berkeley.edu

Abstract

In this paper, we propose a secure federated feature selection protocol, whose core is a secure federated χ^2-test protocol, namely FED-χ^2. In FED-χ^2, we recast χ^2-test, a correlation test, as a second frequency moment estimation problem and use stable projection to encode the local information in a short vector. As such encodings can be aggregated with summation, secure aggregation can be applied to conceal the individual updates. We formally establish the security guarantee of FED-χ^2 by demonstrating that the joint distribution is hidden in a subspace containing exponentially possible distributions. Our evaluation results show that (1) FED-χ^2 achieves good accuracy with small client-side computation overhead; (2) federated feature selection based on FED-χ^2 performs comparably to the centralized feature selection with χ^2-test.

1 Introduction

Federated learning emerged as a new paradigm for large-scale machine learning due to its privacy superiority that the users' raw data never leave their local devices. However, vanilla federated learning lacks formal security guarantee and is shown to be vulnerable to a variety of attacks that infer the users' local data property by observing the users' updates (Nasr, Shokri, and Houmansadr 2019; Pustozerova and Mayer 2020; Chase, Ghosh, and Mahloujifar 2021). To mitigate, Bonawitz et al. (2017) proposed the well-celebrated secure aggregation protocol to only reveal the summed update to the server. Bell et al. (2020) further developed the protocol to improve its communication cost.

Although the security guarantee of FL can be significantly enhanced by secure aggregation, other non-linear learning-related tasks are not compatible with secure aggregation, a representative of which is feature selection. One of feature selection algorithms calculates the correlation score between features and target and keeps the features highly correlated with the target. In the present work, we study a representative correlation test, namely χ^2-test, under the federated setting. There are two straightforward methods for conducting χ^2-test in such a context. First, clients can upload their raw data to the centralized server and delegate the test to it. While this method is effective in terms of communication, it entirely exposes the clients' private information. Second, clients may run secure multiparty computation (MPC) under the server's coordination. Thus, clients can jointly run χ^2-test without disclosing their data to the server. However, general-purpose MPC imposes significant computation and communication overhead, which is typically intolerable in a federated setting with computationally limited clients, *e.g.*, mobile devices.

To address the dilemma, we present a federated protocol optimized for χ^2-test that is *computationally and communicationally efficient* and *discloses limited information to the server*. We begin by recasting χ^2-test as a second frequency moment estimation problem. To approximate the second frequency moment in a federated setting, each client 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 (Bonawitz et al. 2017; Bell et al. 2020) to aggregate the encodings and the server to decode them to approximate the second frequency moment. Because secure aggregation conceals each client's individual update within the aggregated global update, the server learns only limited information about the clients' data.

Our evaluation on four synthetic datasets and 12 real-world datasets shows that FED-χ^2 can replace centralized χ^2-test with good accuracy and low computation overhead. Additionally, we analyze FED-χ^2 in three real-world use cases: feature selection, cryptanalysis, and online false discovery rate control. The results show that FED-χ^2 can achieve comparable performance with centralized χ^2-test and can withstand up to 20% of clients dropping out with minor influence on the accuracy. In summary, we make the following contributions:

- We propose FED-χ^2, the first secure federated χ^2-test protocol. FED-χ^2 is computation- and communication-efficient and leaks much less information than trivially deploying secure aggregation.
- FED-χ^2 decomposes χ^2-test into frequency moments estimation that can easily be encoded/decoded using stable projection and secure aggregation techniques. We give formal security proof and utility analysis of FED-χ^2.
- We evaluate FED-χ^2 in real-world use cases, and the findings suggest that FED-χ^2 can substitute centralized χ^2-test with comparable accuracy, and FED-χ^2 can tolerate up to 20% of clients dropout with minor accuracy drop.

[*]The authors contribute equally to this paper.

Submitted to The Third AAAI Workshop on Privacy-Preserving Artificial Intelligence.

2 Related Work

Bonawitz et al. (2017) proposed the well-celebrated secure aggregation protocol as a low-cost way to 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, Güler, and Avestimehr (2021) proposed TURBOAGG, which combines secure sharing with erasure codes for better dropout tolerance. To improve communication efficiency, Bell et al. (2020) and Choi et al. (2020) 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, Jia, and Song (2020) replaced the binomial noise with discrete Gaussian noise, which is shown to exhibit better composability. Kairouz, Liu, and Steinke (2021) proved that the sum of discrete Gaussian is close to discrete Gaussian, thus discarding the common random seed assumption from Wang, Jia, and Song (2020). The above three works all incorporate secure aggregation in their protocols to lower the noise scale required for differential privacy. Chen et al. (2020) added an extra public parameter to each client to force them to train in the same way, allowing for the detection of malicious clients during aggregation. Nevertheless, designing secure federated correlation tests, despite its importance in real-world scenarios, is not explored by existing research.

On the other end of the spectrum, Wang, Pinelis, and Song (2021) proved that stable projection is differentially private if the projection matrix is secret. In our protocol, the projection matrix is public information; hence FED-χ^2 does not consider the differential privacy guarantee.

3 Federated Correlation Test

In this section, we elaborate on the design of FED-χ^2, a secure federated protocol for χ^2-test. We first formalize the problem, establish the notation system, and introduce the threat model. Then we recast χ^2-test as a second frequency moment estimation problem in the federated setting, and consequently, we are able to leverage stable projection to encode each client's local information, and then aggregate them using secure aggregation. At last, we present security proof, utility analysis, communication analysis, and computation analysis of FED-χ^2.

3.1 Problem Setup

We now formulate the federated correlation test and establish the notation system. 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.*, $\mathbf{A}, \mathbf{B}, \mathbf{C}$).

We consider a population of n clients $\mathcal{C} = \{c_i\}_{i\in[n]}$. Each client has one share of local data composed of the triplets $\mathcal{D}_i = \{(x, y, v_{xy}^{(i)})\}, x \in \mathcal{X}, y \in \mathcal{Y}, v_{xy}^{(i)} \in \{-M, \cdots, M\}$, where x and y are categories of the contingency table, $v_{xy}^{(i)}$ is the observed counting of the categories x and y in the local contingency table of the i^{th} client, $|\mathcal{X}| = m_x$ and $|\mathcal{Y}| = m_y$ are finite domains, and M is the maximum value $|v_{xy}^{(i)}|$ can be. The global dataset is defined as $\mathcal{D} = \{(x, y, v_{xy}) : v_{xy} = \sum_{i\in[n]} v_{xy}^{(i)}\}$. We focus on federated χ^2-test and the data in contingency table is discrete. For the ease of presentation, we define the marginal statistics $v_x = \sum_{y\in[|\mathcal{Y}|]} v_{xy}, v_y = \sum_{x\in[|\mathcal{X}|]} v_{xy}$, and $v = \sum_{x\in[|\mathcal{X}|], y\in[|\mathcal{Y}|]} v_{xy}$. Besides, we define $\bar{v}_{xy} = \frac{v_x \times v_y}{v}$, denoting the expectation of v_{xy} if x and y are uncorrelated. We define $m = m_x m_y$ and use an indexing function $\mathbb{I} : [m_x] \times [m_y] \to [m]$ to obtain a uniform indexing given the indexing of each variable. A centralized server $\mathcal{S}$ calculates the statistics for χ^2-test $s_{\chi^2}(\mathcal{D}) = \sum_{x\in[|\mathcal{X}|], y\in[|\mathcal{Y}|]} \frac{(v_{xy} - \bar{v}_{xy})^2}{\bar{v}_{xy}}$ on the global dataset to decide whether $\mathcal{X}$ and $\mathcal{Y}$ are correlated without collecting the raw data from clients.

Overall, using MPC to conduct secure correlation tests in a federated scenario is highly expensive and impractical (Boyle, Chung, and Pass 2015; Damgård et al. 2012). Hence, in the present work, we trade off accuracy for efficiency, as long as the estimation error is small with a high probability. Formally, if FED-χ^2 outputs $\hat{s}_{\chi^2}$, whose corresponding standard centralized χ^2-test output is s_{χ^2}, the following accuracy requirement should be satisfied with small ϵ and δ.

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

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, client updates should contain as little sensitive information as feasible. We want to emphasize that, while the server may explore the privacy of clients, the server will honestly follow the protocol due to regulation or reputational pressure. The server won't provide adversarial vectors to the clients.

On the other hand, we assume honest clients. 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.

3.2 From Correlation Test to Frequency Moments Estimation

We first recast correlation test to a second frequency moments estimation problem as defined below. Given a set of key-value pairs $\mathcal{S} = \{k_i, v_i\}_{i\in[n]}$, we re-organize it into a histogram $\mathcal{H} = \{k_j, v_j = \sum_{k_i=k_j, i\in[n]} v_i\}$, and estimate the α^{th} frequency moments as $F_\alpha = \sum_j v_j^\alpha$. χ^2-test can thus be recast to a 2^{nd} frequency moments estimation problem:

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

In federated setting, each client c_i holds a local dataset $\mathcal{D}_i = \{(x, y, v_{xy}^{(i)})\}$ and computes a vector $\mathbf{u}_i$, where $\mathbf{u}_i[\mathbb{I}(x, y)] = \frac{v_{xy}^{(i)} - \bar{v}_{xy}/n}{\sqrt{\bar{v}_{xy}}}$ and $\mathbf{u}_i$ has m elements. Thus, the challenge in federated χ^2-test becomes calculating the following equation:

$$s_{\chi^2}(\mathcal{D}) = \sum_{x,y} (\frac{v_{xy} - \bar{v}_{xy}}{\sqrt{\bar{v}_{xy}}})^2 = || \sum_{i\in[n]} \mathbf{u}_i||_2^2$$

3.3 Encoding by Stable Projection & Decoding by Geometric Mean Estimator

To address the aforementioned challenges and to easily integrate the algorithm into secure aggregation protocols, we use *stable random projection* (Indyk 2006; Vempala 2005) and *geometric mean estimator* (Li 2008) to approximate the data's second frequency moment efficiently. We begin by discussing stable distributions, followed by the encoding and decoding techniques.

Definition 1 (α-stable distribution)**.** *A random variable X follows an α-stable distribution $\mathcal{Q}_{\alpha,\beta,F}$ if its characteristic function is as follows.*

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

, where F is the scale to the α^{th} power 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 2 (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 $C_1X + C_2Y \sim \mathcal{Q}_{\alpha,\beta,C_1^\alpha + C_2^\alpha}$.*

We borrow the genius idea from Indyk's well-celebrated paper (Indyk 2006) to encode the frequency moments in the scale parameter of a stable distribution defined in Definition 1. To encode the local dataset $\mathcal{D}_i = \{(x, y, v_{xy}^{(i)})\}$, each client c_i, $i \in [n]$, is given by the server an $\ell \times m$ projection matrix $\mathbf{P}$ whose values are drawn independently from an α-stable distribution $\mathcal{Q}_{\alpha,\beta,1}$, where ℓ is the encoding size and $m = m_x m_y$. The client re-organizes the data into a vector $\mathbf{u}_i$, where $\mathbf{u}_i[\mathbb{I}(x, y)] = v_{xy}^{(i)}$. Then, the client projects $\mathbf{u}_i$ to $\mathbf{e}_i = \mathbf{P} \times \mathbf{u}_i$ as the encoding (line 2 in Alg. 1).

To decode, the server first sums the encodings up $\mathbf{e} = \sum_{i\in[n]} \mathbf{e}_i$ and estimates the scale of the variables in the aggregated encoding with an unbiased geometric mean estimator (Li 2008) in line 4 of Alg. 1. According to the α-stability defined in Definition 2, every element e_k in $\mathbf{e}$, $k \in [\ell]$, follows this stable distribution: $e_k \sim \mathcal{Q}_{2,0,||\sum_{i\in[n]}\mathbf{u}_i||_2^2}$. Thus, the l_2 norm can be estimated by calculating the scale of the distribution $\mathcal{Q}_{2,0,||\sum_{i\in[n]}\mathbf{u}_i||_2^2}$ with $\mathbf{e}$ containing ℓ elements.

During the above processes, $\mathbf{u}_i$ and $\sum_{i\in[n]}\mathbf{u}_i$ are not leaked and the aggregation among clients is calculated with only summation. Thus, secure aggregation protocols can be naturally applied with quantization. Furthermore, $\ell = \frac{c}{\epsilon^2}\log(1/\delta)$ suffices to guarantee that the second frequency moment can be approximated with a $1 \pm \epsilon$ factor and a probability no less than $1 - \delta$. We will further analyze the utility of FED-χ^2 in Sec. 3.6.

Algorithm 1: The encoding and decoding scheme for federated frequency moments estimation. Note that the encoding and decoding themselves do not provide any security guarantee.

```
1 Function ENCODE (P, u_i):
2     return $\mathbf{P} \times \mathbf{u}_i$
3 Function GEOMETRICMEANESTIMATOR (e):
4     $\hat{d}_{(2),gm} \leftarrow \frac{\prod_{k=1}^{\ell}|\mathbf{e}_k|^{2/\ell}}{(\frac{2}{\pi}\Gamma(\frac{2}{\ell})\Gamma(1-\frac{1}{\ell})\sin(\frac{\pi}{\ell}))^\ell}$    // $\ell$ is the
       encoding size.
5     return $\hat{d}_{(2),gm}$
6 Function DECODE (e):
7     return GEOMETRICMEANESTIMATOR (e)
```

3.4 Secure Federated Correlation Test

The complete protocol for FED-χ^2 is presented in Alg. 2. Firstly, the marginal statistics v_x, v_y and v are calculated with secure aggregation and broadcasted to all clients (lines 1–6 in Alg. 2). This step can be omitted if the marginal statistics are already known. Then, on the server side, a projection matrix $\mathbf{P}$ is sampled from an α-stable distribution $\mathcal{Q}_{2,0,1}^{\ell\times m}$. The projection matrix is broadcasted to all clients (lines 8–10 in Alg. 2). For each client c_i, the local data is re-organized into $\mathbf{u}_i$ and projected to $\mathbf{e}_i$ as encoding (lines 11–14 in Alg. 2). Then, the encoding results will be quantized and aggregated with secure aggregation (line 15 in 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, since the size of the contingency table is normally moderate rather than enormous. Thus, the precision of the quantized float number is comparable to or even better than that of float64, and hence we disregard the effect of quantization on accuracy. Finally, the server gets the χ^2-test result using the decoding algorithm described in Alg. 1 (line 17 in Alg. 2).

Dropouts in the first round have no effect on the test's accuracy because they can be recovered inside secure aggregation (Bonawitz et al. 2017; Bell et al. 2020). On the other hand, dropouts in the second round will affect the accuracy of the test. Still, because 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).

3.5 Security Analysis

In this paper, we choose the state-of-the-art secure aggregation protocol by Bell et al. (Bell et al. 2020), which replaces the complete graph with a sparse random graph to enhance communication efficiency. We clarify that FED-χ^2 can incorporate other popular secure aggregation protocols. 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 C 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 $\{v_x\}, \{v_y\}$, and all the encodings $\{\boldsymbol{e}_i\}$, the output of* SIM *is indistinguishable from the view of the real server $\Pi_{\mathcal{C}}$ in that execution,* i.e., *$\Pi_{\mathcal{C}} \approx_\lambda$ SIM$(\sum \boldsymbol{e}_i, n)$.*

Intuitively, Theorem 1 illustrates that no more information about the clients except the averaged updates is revealed to the centralized server. Thus, each client's update is hidden by the rest clients in secure aggregation. We now present the formal proof for Theorem 1.

Algorithm 2: FED-χ^2: secure federated χ^2-test. 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 is the client number.
    for $x \in [m_x]$ do $v_x = $ SecureAgg ($\{v_x^{(i)}\}_{i\in[n]}$)
    for $y \in [m_y]$ do $v_y = $ 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
    Server
        Sample the projection matrix $\mathbf{P}$ from $\mathcal{Q}_{2,0,1}^{\ell\times m}$
        Broadcast the projection matrices to the clients
    Client $c_i, i \in [n]$
        Calculate $\bar{v}_{xy} = \frac{v_x v_y}{v}$
        Prepare $\mathbf{u}_i$ s.t. $\mathbf{u}_i[\mathbb{I}(x,y)] = \frac{v_{xy}^{(i)} - \bar{v}_{xy}/n}{\sqrt{\bar{v}_{xy}}}$
        Calculate $\mathbf{e}_i = $ Encode ($\boldsymbol{P}, \boldsymbol{u}_i$)
    $\mathbf{e} = $ SecureAgg (Quantize ($\{\boldsymbol{e}_i\}_{i\in[n]}$))
    Server
        $\hat{s}_{\chi^2} = $ Decode ($\boldsymbol{e}$)
```

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 Appendix C 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 all inputs $\mathcal{X} = \{\boldsymbol{e}_i\}_{i\in[n]}$, the output of* SIMSA *is perfectly indistinguishable from the view of the real server,* i.e., $\text{SECUREAGG}_{\mathcal{C}} \approx_\lambda \text{SIMSA}(\sum_{i\in[n]} \boldsymbol{e}_i, n)$.

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 χ^2-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, REAL$_{\mathcal{C}}$.

HYB$_2$ In this hybrid, we replace the view during the execution of each SECUREAGG($\{v_x^{(i)}\}_{i\in[n]}$) in line 3 of Alg. 2 with the output of SIMSA(v_x, n) one by one. 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 SECUREAGG($\{v_y^{(i)}\}_{i\in[n]}$) in line 4 of Alg. 2 with the output of SIMSA(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 SECUREAGG($\{\mathbf{e}_i\}_{i\in[n]}$) in line 15 of Alg. 2 with the SIMSA($\sum \mathbf{e}_i, n$). This hybrid is the output of SIM. According to Lemma 1, HYB$_4$ is indistinguishable from HYB$_3$. □

Remark: what does Alg. 2 leak? 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:

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

, 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 a vector flattened by $\mathbf{V}$. To understand this, information leaked by FED-χ^2 includes the estimation $\mathbf{e}$ and marginal statistics $\mathbf{v}_x$ and $\mathbf{v}_y$. The following theorem establishes an important fact: the above equation system has an exponentially large solution space, which effectively conceals the real joint distribution. We thus believe that Alg. 2 practically ensures privacy due to the solution space's vastness.

Theorem 2. *Given a projection matrix $\boldsymbol{P} \in \mathbb{Z}_q^{\ell\times m}$, $\{v_x\}$, $\{v_y\}$ and $\boldsymbol{e}$, there are at least $q^{m-\ell-m_x-m_y}$ feasible choices of $\{v_{xy}\}$.*

Proof sketch for Theorem 2. As demonstrated above, the given information forms a linear equation with $m_x + m_y + \ell$ equations. Given $m > m_x + m_y + \ell$, the rank of the coefficient matrix is no more than $m_x + m_y + \ell$. Solving the equations with Smith normal form, we know that the solution space is at least $(m - m_x - m_y - \ell)$-dimensional. With the following lemma, we manage to prove Theorem 2.

Lemma 2. *There are $q^{r\times c}$ vectors in the subspace of $\mathbb{Z}_q^{r\times c}$.* □

3.6 Utility Analysis

In this section, we conduct the utility analysis in terms of multiplicative error. 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.

Theorem 3 (Utility)**.** *Let* Π *be an instantiation of Alg. 2 with secure aggregation protocol in Alg. 4 of Appendix C.* Π *is parameterized with $\ell = \frac{c}{\epsilon^2}\log(1/\delta)$ for some constant c. 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:*

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

Proof sketch for Theorem 3. First, we introduce the following lemma from Li (2008).

Lemma 3 (Tail bounds of geometric mean estimator (Li 2008))**.** *The right tail bound of geometric mean estimator is:*

$$\mathbb{P}(\hat{s}_{\chi^2} - s_{\chi^2} > \epsilon s_{\chi^2}) \leq \exp(-\ell \frac{\epsilon^2}{G_R})$$

, where $\frac{\epsilon^2}{G_R} = C_1 \log(1 + \epsilon) - C_1 \gamma_e(\alpha - 1) - \log(\frac{2}{\pi}\Gamma(\alpha C_1)\Gamma(1 - C_1)\sin(\frac{\pi\alpha C_1}{2}))$*,* $\alpha = 2$ *in our setting,* $C_1 = \frac{2}{\pi}\tan^{-1}(\frac{\log(1+\epsilon)}{(2+\alpha^2)\pi/6})$*, and* $\gamma_e = 0.577215665...$ *is the Euler's constant.*

The left tail bound of the geometric mean estimator is:

$$\mathbb{P}(\hat{s}_{\chi^2} - s_{\chi^2} < -\epsilon s_{\chi^2}) \leq \exp(-\ell \frac{\epsilon^2}{G_L})$$

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

With Lemma 3, Taking $c \geq \max(G_R, G_L)$ and $\delta = \exp(-\frac{\ell\epsilon^2}{c})$, we are able to prove $\mathbb{P}[\hat{s}_{\chi^2} < (1 - \epsilon)s_{\chi^2} \vee \hat{s}_{\chi^2} > (1 + \epsilon)s_{\chi^2}] \leq \delta$, and the above bound holds when $\ell = \frac{c}{\epsilon^2}\log(1/\delta)$. □

3.7 Communication & Computation Analysis

In this section, we present the communication and computation cost of Alg. 2.

Theorem 4 (Communication Cost)**.** *Let* Π *be an instantiation of Alg. 2 with secure aggregation protocol in Alg. 4 of Appendix C, 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 5 (Computation Cost)**.** *Let* Π *be an instantiation of Alg. 2 with secure aggregation protocol in Alg. 4 of Appendix C, 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* $\mathcal{O}(n\log^2 n + n(\ell + m_x + m_y)\log n + \ell)$*.*

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 D for the detailed proof of Theorem 4 and Theorem 5.

4 Evaluation

Experiment Setup. To assess FED-χ^2's accuracy, we simulate it on four synthetic datasets and 12 real-world datasets. We compare the multiplicative errors 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. As the real-world datasets, we report the details in Appendix E.

Additionally, we evaluate FED-χ^2's utility in federated feature selection and another two applications of FED-χ^2: cryptanalysis and online FDR control. For feature selection, we report the model accuracy trained on the selected features. For cryptanalysis, we report the success rate of cracking ciphertexts. 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 three experiments. 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 Evaluation Results

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 is independent of the number of clients.

From Fig. 1, we observe that the larger the encoding size ℓ, the smaller the multiplicative error. When $\ell = 50$, the multiplicative error $\epsilon \approx 0.2$. This conforms with Theorem 3, in which the multiplicative error $\epsilon = \sqrt{\frac{c}{\ell}\log(1/\delta)}$ decreases as ℓ increases.

We also evaluate the power (Cohen 2013) of FED-χ^2. We set the p-value threshold as 0.05, which determines whether or not to reject the null hypothesis. 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. Specifically, since the χ^2 values are typically far from the decision threshold, a multiplicative error of 0.2 rarely flips the final decision.

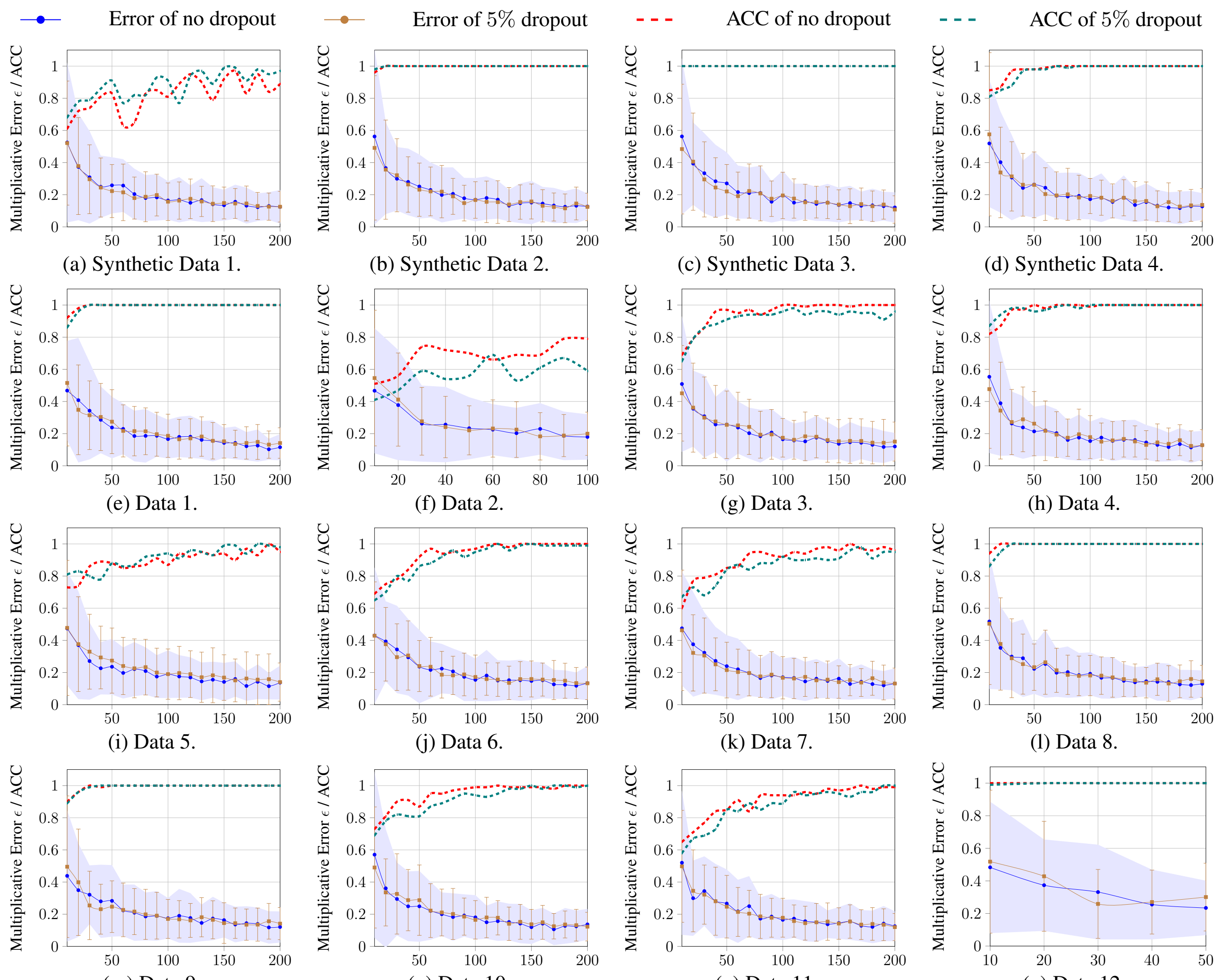
We also present the results when 5% of clients drop out in the second round of FED-χ^2 in Fig. 1. The results show that FED-χ^2 is robust to a small portion of dropouts. In Appendix B, we present the results in terms of 10%, 15%, and 20% dropout rates.

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.

The results are shown in Fig. 2. Each point represents the average of 100 separate runs, with accompanying error bars. The overhead is 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 equals m_y.

4.2 Feature Selection.

We first evaluates 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 and Weinstein 1990), a standard text categorization dataset (Yang 1999; Yang and Pedersen 1997; Zhang and Yang 2003), and pick the top-20 most frequent categories using 17,262 training and 4,316 test documents. These documents are distributed randomly to 100 clients, each of whom receives the same number of training documents. After removing all numbers



(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.

Figure 1: Multiplicative error and accuracy of FED-χ^2 w.r.t. encoding size ℓ w/ and w/o dropout.

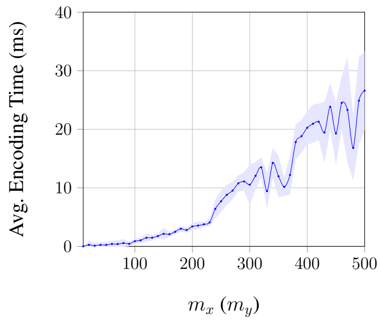


Figure 2: Client-side encoding overhead when $m_x = m_y$.

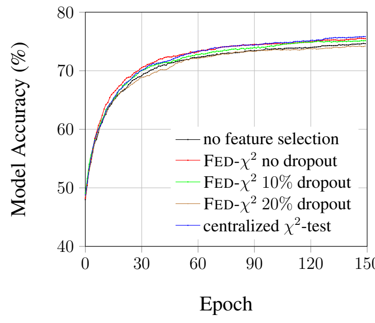


Figure 3: Accuracy of the model trained with features selected by FED-χ^2 and centralized χ^2-test.

and stop-words, we obtain 167,135 indexing terms. 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 F. 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.

4.3 Other Downstream applications.

Cryptanalysis. In the second case study, we explore federated cryptanalysis with FED-χ^2. We break Caesar cipher (Luciano and Prichett 1987), a classic substitution cipher, with FED-χ^2. In a Caesar cipher, each letter in the plaintext is replaced by another letter with some fixed number of positions down the alphabet. For instance, each English letter can be right-shifted by three, converting the plaintext "good" to the ciphertext "jrrg". There are 26 possible shifts when given 26 English letters. The plaintext can be cracked in a shortcut by performing a correlation test on the ciphertext in relation to normal English text.

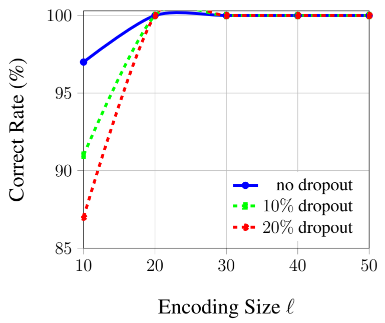


Figure 4: The success rate of cracking Caesar ciphers.

In our setting, each client is assumed to possess a segment of the Caesar ciphertext. To collaboratively crack the ciphertext, these clients run 26 χ^2-tests to determine the correlation level between each ciphertext letter and the letters in normal English text. The χ^2-test yielding the highest correlation level elucidates how English letters are encrypted into Caesar ciphertexts.

We take Shakespeare's lines as the plaintext and encrypt it into a Caesar ciphertext with a length of 1000 characters. We initiate the cracking process on ten non-overlapping ciphertexts to compute the average success rate (see Fig. 4). In general, the larger the encoding size, the more precise the χ^2-statistics is, and consequently the higher the success rate. Again, according to Theorem 3, the multiplicative error ϵ decreases as the encoding size increases. In Fig. 4, we also report the success rate when 10% and 20% of clients dropout in Round two of FED-χ^2, respectively. Even if 20% of clients dropout, the success rate can still be 100% as long as the encoding size ℓ is larger than 20.

Online False Discovery Rate Control. In the third case study, we explore federated online false discovery rate (FDR) control (Foster and 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

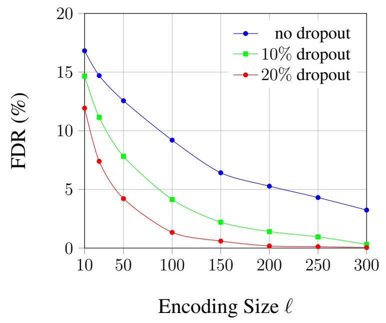


Figure 5: Average FDR w.r.t. ℓ for SAFFRON with FED-χ^2.

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 detailed algorithm of SAFFRON and the hyper-parameters used in our evaluation in Appendix G. The size of the randomly synthesized contingency table is 20×20. 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. 5. We observe that the variance of independent runs is very small, so we omit the error bars. The results indicate that by increasing the encoding size ℓ, FED-χ^2 can achieve a low FDR of less than 5.0%. We also observe that dropouts improve the performance of FED-χ^2 in this case study. The reason for this is because dropouts reduce the estimated χ^2 value, which increases the probability of accepting the null hypothesis in FED-χ^2. As a larger portion of queries follows the null hypothesis in online FDR control, the accuracy also increases. The results further demonstrate that FED-χ^2 can be employed in practice to facilitate online FDR control using a secure federated correlation test.

5 Conclusion

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 an important application of the protocol: χ^2-test. We give formal security proof and utility analysis on our proposed secure federated learning χ^2-test protocol FED-χ^2 and validate them with empirical evaluations and case studies.

References

Agarwal, N.; Suresh, A. T.; Yu, F.; Kumar, S.; and Mcmahan, H. B. 2018. cpSGD: Communication-efficient and differentially-private distributed SGD. *arXiv preprint arXiv:1805.10559*.

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

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

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

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

Boyle, E.; Chung, K.-M.; and Pass, R. 2015. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In *Annual Cryptology Conference*, 742–762. Springer.

Chase, M.; Ghosh, E.; and Mahloujifar, S. 2021. Property Inference From Poisoning. *arXiv preprint arXiv:2101.11073*.

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

Choi, B.; Sohn, J.-y.; Han, D.-J.; and Moon, J. 2020. Communication-Computation Efficient Secure Aggregation for Federated Learning. *arXiv preprint arXiv:2012.05433*.

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

Damgård, I.; Pastro, V.; Smart, N.; and Zakarias, S. 2012. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, 643–662. Springer.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Foster, D. P.; and Stine, R. A. 2008. 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.

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

Hayes, P. J.; and Weinstein, S. P. 1990. CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. In *IAAI*, volume 90, 49–64.

Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; and Igel, C. 2013. Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 1288.

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

Kairouz, P.; Liu, Z.; and Steinke, T. 2021. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. *arXiv preprint arXiv:2102.06387*.

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

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

Li, P. 2008. 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*, 10–19.

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

Luciano, D.; and Prichett, G. 1987. Cryptology: From Caesar ciphers to public-key cryptosystems. *The College Mathematics Journal*, 18(1): 2–17.

Nasr, M.; Shokri, R.; and Houmansadr, A. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, 739–753. IEEE.

Pustozerova, A.; and Mayer, R. 2020. Information leaks in federated learning. In *Proceedings of the Network and Distributed System Security Symposium*.

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

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

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

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

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

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

Wang, L.; Jia, R.; and Song, D. 2020. D2P-Fed: Differentially private federated learning with efficient communication. *arxiv. org/pdf/2006.13039*.

Wang, L.; Pinelis, I.; and Song, D. 2021. Differentially Private Fractional Frequency Moments Estimation with Polylogarithmic Space. *arXiv preprint arXiv:2105.12363*.

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

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

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

Zhang, J.; and Yang, Y. 2003. 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*, 190–197.

Appendix

A Clarification on Privacy

As indicated in Sec. 3, the projection matrix is public information, and hence FED-χ^2 does not take the differential privacy guarantee into account. We would want to provide more information in order to eliminate any potential misunderstandings.

To begin, we would want to emphasize that "privacy" in our paper refers to MPC-style privacy, not DP-style privacy. In general, MPC-style privacy is orthogonal to DP-style privacy: in MPC, privacy is obtained against a semi-honest server in such a way that the server cannot witness individual client's updates but only an aggregate of them, e.g. SecAgg (Bonawitz et al. 2017). In DP, privacy is accomplished by including random noise in each client's update, such that the distribution of the output result does not reveal the clients' private information and the server cannot infer the clients' identification from the output result.

Second, we would like to emphasize that our work proposes a novel secure aggregation scheme particularly for the χ^2-test. Existing standard secure aggregation schemes are inapplicable to the χ^2-test, which will reveal much more information than FED-χ^2, as we have clarified in Sec. 1. Again, this work requires guaranteeing MPC-style privacy, not DP.

Third, to quantify MPC-style privacy, we prove in Theorem 2 that the clients' updates in FED-χ^2 are hidden inside a space with exponential size. This is weaker than hiding users' updates in the whole space, but still gives meaningful privacy guarantees (consider attempting to guess the output of an exponential-sided dice, which is practically infeasible).

Finally, while DP is orthogonal to this research, we would want to emphasize that our protocol can achieve DP by introducing calibrated discrete Gaussian noise to the users' local updates.

B Further Results on FED-χ^2 with Dropouts

We present the results of 10%, 15%, and 20% clients dropout in Fig. 6. The results further show that FED-χ^2 can tolerate a considerable portion of clients dropout in Round 2 of Alg. 2.

C Secure Aggregation

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 3 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.

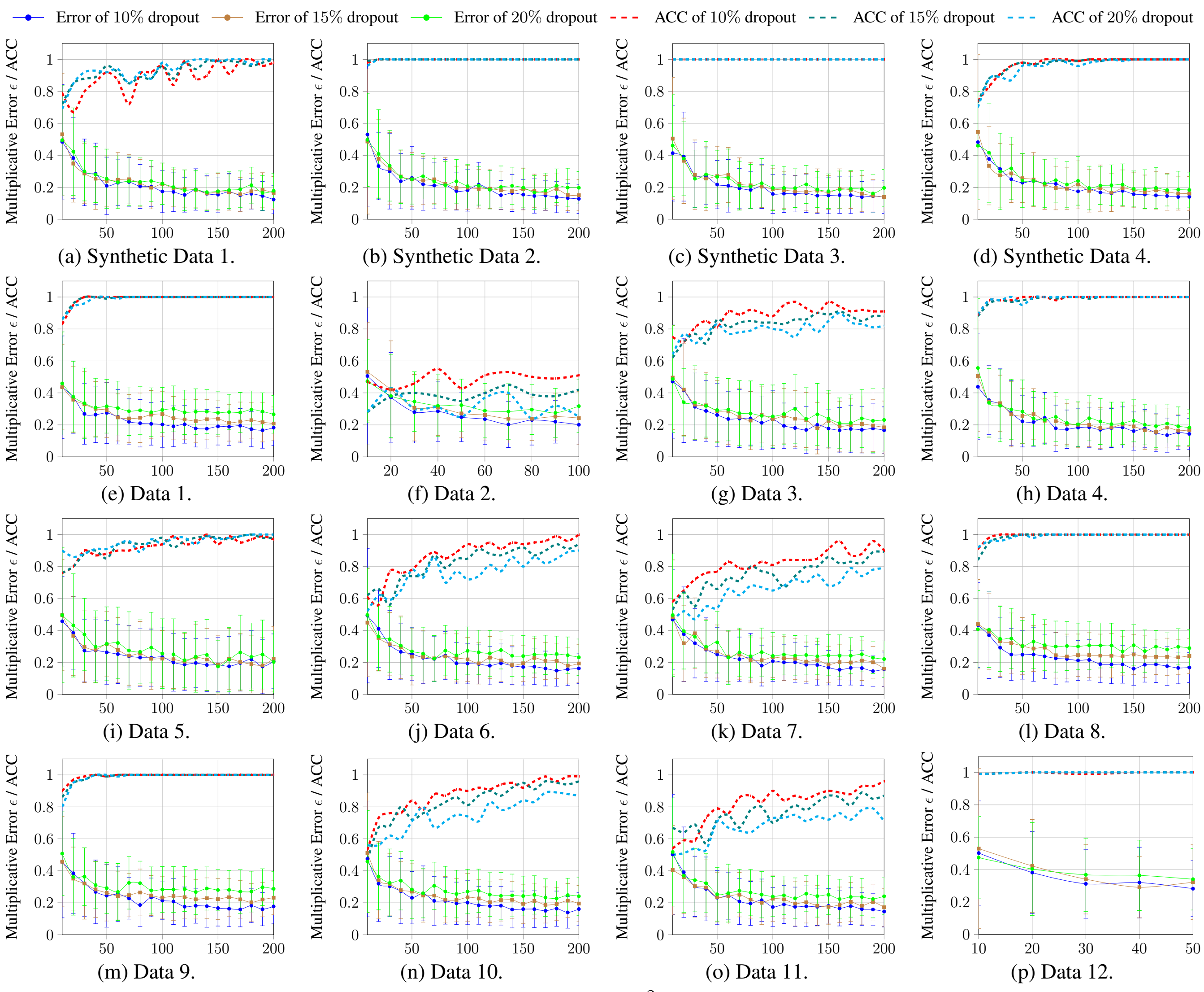
In the fifth step, each pair (i, j) of connected clients in G runs a λ-secure key agreement protocol $s_{i,j} = \mathcal{KA}.Agree(sk_i^1, pk_j^1)$ which uses the key exchange in the previous step to derive a shared random key $s_{i,j}$. The pairwise masks $\mathbf{m}_{i,j} = F(s_{i,j})$ can be computed, 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: $\sum_{i \in A_2'} (\mathbf{y}_i - \mathbf{r}_i + \sum_{j \in NG(i) \cap (A_1' \setminus A_2'), 0<j<i} \mathbf{m}_{i,j} - \sum_{j \in NG(i) \cap (A_1' \setminus A_2'), i<j \le n} \mathbf{m}_{i,j})$.

Definition 3 (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* $j - i \pmod n \le (k+1)/2$ *or* $j - i \pmod n \ge n - k/2$.

D Proof for Communication & Computation Cost

We provide the proof for Theorem 4 and Theorem 5 in the following.



(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.

Figure 6: Multiplicative error and accuracy of FED-χ^2 w.r.t. encoding size ℓ w/ and w/o dropout.

Algorithm 3: INITSECUREAGG: Generate Initial Graph for SECUREAGG.

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

Theorem 4 (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 4. 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 5 (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 5. Each client computation can be broken up as k key agreements ($\mathcal{O}(k)$ complexity, line

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

```
1  Function SecureAgg($\{e_i\}_{i\in[n]}$):
2      ▷ Parties: Clients $c_1, \cdots, c_n$, and Server.
3      ▷ $l$: Vector length.
4      ▷ $\mathbb{X}^l$: Input domain, $\mathbf{e}_i \in \mathbb{X}^l$.
5      ▷ $F : \{0,1\}^\lambda \to \mathbb{X}^l$: PRG.
6      ▷ 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.
7      (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).
8      (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 $N_G(i)$.
9      (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 $j \in N_G(i)$.
10     (4) The server aborts if $|A'_1| < (1-\delta)n$ and otherwise forwards $(j, c_{i,j})$ to client $c_j$ who deduces $A'_1 \cap N_G(j)$.
11     (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
               $\mathbf{y}_i = \mathbf{e}_i + \mathbf{r}_i - \sum_{j\in[n], j<i} \mathbf{m}_{i,j} + \sum_{j\in[n], j>i} \mathbf{m}_{i,j}$
12     (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 $c_i, i \in A'_2$.
13     (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).
14     (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 $\sum_{i\in A'_2}(\mathbf{y}_i - \mathbf{r}_i + \sum_{j\in NG(i)\cap(A'_1\backslash A'_2), 0<j<i} \mathbf{m}_{i,j} - \sum_{j\in NG(i)\cap(A'_1\backslash A'_2), i<j\leq n} \mathbf{m}_{i,j})$.
```

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), 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. □

E 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 2021) 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 and Graff 2017).

F Details of Regression Models

The details of the regression models trained in feature selection in Sec. 4.3 is reported in Table 2. The training and

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 2021)	Occupation	14	Native Country	41
2	Credit Risk Classification (Govindaraj, Praveen 2021)	Feature 6	14	Feature 7	11
3	Credit Risk Classification (Govindaraj, Praveen 2021)	Credit Product Type	28	Overdue Type I	35
4	Credit Risk Classification (Govindaraj, Praveen 2021)	Credit Product Type	28	Overdue Type II	35
5	Credit Risk Classification (Govindaraj, Praveen 2021)	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 2021)	Cap color	10	Odor	9

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

G SAFFRON Procedure

In Sec. 4.3, 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 j^{th} 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 j^{th} 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 we use for the SAFFRON procedure in online false discovery rate control of Sec. 4 are aligned with the setting in (Ramdas et al. 2018). In particular, 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: $\gamma_j = \frac{1/(j+1)^{1.6}}{\sum_{j=0}^{10000} 1/(j+1)^{1.6}}$.

Algorithm 5: SAFFRON Procedure.

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