Privacy-Preserving and Trustless Verifiable Fairness Audit of Machine Learning Models

Gui Tang[1], Wuzheng Tan[2], Mei Cai[3],
College of Cyber Security, Jinan University, Guangzhou, China[1,2]
Jinan University Library, Guangzhou, China[3]

Abstract—**In the big data era, machine learning has developed prominently and is widely used in real-world systems. Yet, machine learning raises fairness concerns, which incurs discrimination against groups determined by sensitive attributes such as gender and race. Many researchers have focused on developing fairness audit technique of machine learning model that enable users to protect themselves from discrimination. Existing solutions, however, rely on additional external trust assumptions, either on third-party entities or external components, that significantly lower the security. In this study, we propose a trustless verifiable fairness audit framework that assesses the fairness of ML algorithms while addressing potential security issues such as data privacy, model secrecy, and trustworthiness. With succinctness and non-interactive of zero knowledge proof, our framework not only guarantees audit integrity, but also clearly enhance security, enabling fair ML models to be publicly auditable and any client to verify audit results without extra trust assumption. Our evaluation on various machine learning models and real-world datasets shows that our framework achieves practical performance.**

Keywords—Security and privacy; machine learning; fairness; cryptography; zero knowledge proof

I. Introduction

Machine Learning has seen great success in decision-making and decision-support tasks in recent years [1] [2] [3], being deployed in various applications and products in practice, such as loans and hiring decisions. However, concerns are rising that algorithms amplify bias and discrimination from the training data, and fairness is becoming an essential metric for evaluating machine learning models [4]–[8]. Consequently, fairness has become a roadblock to widespread machine learning applications. To address this formally, many works towards considering how algorithm fairness can be assessed by proposing various measures and how discrimination in machine learning systems can be mitigated by pre-processing [9], [10], inter-processing [11], [12], and post-processing methods [5] [13].

In practice, there is a need for guarantees that the result of fairness audit are correctly calculated with respect to specific fairness metrics, which is referred to as the *audit integrity* of fairness. One of the basic ideas to ensure that users are protected from discrimination is to ensure the integrity of the audit. In order to get a fair model, the server usually requires the user's sensitive data, such as gender and race, to train the machine learning model. However, this requirement is often contrary to the interests of the user. First, users are usually reluctant to share their data, even if it is a reasonable aim, because it would expand their exposure to privacy risks. In addition, the collection of sensitive user data is subject to legal restrictions. For example, the EU's General Data Protection Regulation (GDPR) highlights the minimal prerequisites for collecting sensitive data [14]. If the model itself is not a secret, anyone can potentially run tests on it to establish its purported fairness without exposing its data. However, this approach may be contrary to the benefits of the model owner due to intellectual property. Although there are fair learning approaches [15], [16], training fair models without the sensitive data have been proposed, it is still required to have the sensitive data for assessing the fairness of the trained model [17]. We call this problem as *sensitive data availability*.

To overcome sensitive data availability issues in providing audit integrity, Veale and Binns [14] introduce a trusted third party with sensitive data to certify the fairness of a machine learning model. Although this model works well, it requires a strong trust relationship between the third party and the model owner. Either the third party has access to the ML model, or the model owner has access to the sensitive data, which may be against their interests. To audit the model publicly while protecting sensitive data's privacy and keeping the model confidentiality, Kilbertus et al. [18] and Segal et al. [19] proposed to utilize multi-party computation (MPC) approach. Those approaches enable a public fairness audit under the assumption of a semi-honest security model and are extended by Pentyala et al. [20] to a malicious security model. Park et al. [17] propose a framework to enable secure fairness audit by leveraging confidential computing based on hardware enclave under the malicious security model.

The problem. While previous work have been work well, existing solutions still suffer from extra trust assumption. This is problematic for two reasons. First, the additional trust assumptions mean the third party determines the audit integrity. Second, relying on a third party can lead to single-point failures. Specifically, the MPC-based approach assumes that the server and the third party are running all required steps in a protocol. Moreover, the hardware-based approach introduces additional hardware security assumptions and also suffers from hardware vulnerabilities [21] [22]. Furthermore, when we are in a situation where we want to audit the model used for several different domains, we need to establish credible relationships with more third parties or hardware enclaves.

To get secure and robust service, we need a much more robust security guarantee: each party only trusts itself. This raises our question: *Can we design a framework for auditing the fairness of machine learning models under no trusted party existing scenario? Or can we guarantee security for audit*

[2]Corresponding Author.

integrity without external trust assumptions? For example, we want to support fairness audits as a service in the model market to achieve fairness integrity.

We answer the question above positively in this paper by proposing a fair audit framework, which enables a publicly verifiable fairness audit of the ML model without disclosing model parameters and guarantees audit integrity of the fair audit. The main idea is to leverage the progress of zero knowledge succinct non-interactive arguments of knowledge (zk-SNAKRs) [23]–[29] recently. A zk-SNARK enables the third party to efficiently convince the verifier that the computation of fairness audit is correctly calculated. We solve the critical challenge of adapting zk-SNARK to this work under the malicious threat model. In summary, the contributions of this work are:

- We provide a generic framework to audit the fairness of machine learning model under the trustless condition. We can support generic machine learning models with arbitrary fairness metrics.
- We formally define security requirements and instantiate the framework described above. We have solved performance challenging problems in the instantiation process.
- We implement our framework and evaluate its performance on several real-world datasets. The experimental results show that our framework achieves practical performance.

II. Literature Review

A. Fairness Audit

Despite training a machine learning model is a fundamental problem, bringing the model to reality is also important. A fundamental question is how to ensure that the model used is non-discriminatory. There is a line of work to discuss this problem. Veale and Binns [14] introduced highly trusted third parties selectively storing data and performing discrimination auditing to achieve fairness in machine learning. However, they assume the modeler must disclose their model to a third party or trust it in order to obtain the model prediction on test data, which may be incompatible with modeler's intellectual property. To resolve these problems, other privacy-preserving approaches such as multi-party computation or trusted execute environment can be applied. Kibertus et al. [18] and Segal et al. [19] proposed privacy-preserving fair certification and inference of ML model that protect sensitive attributes and model confidentiality by using MPC. However, they assume that two honest-but-curious server and require high communication. The following work PrivFair [20] extend their security model to active security threat models in 2- or 3-server setups. Park et al. [17] provided a generic fairness audit framework that relies on hardware enclaves and explores more potential threats and attacks in the fairness certification process. Although their approach has a small computational overhead, their require additional hardware and trustworthiness to TEE, which is not our goal. And TEE also face many unknown vulnerability [21]. All of these work require additional trusts, and does not provide public verifiable. There computation integrity rely on their trust on third party. In this study, we explore a publicly verifiable security audit protocol based on zero knowledge proof with lower level of trust. Also [19] and [17] explore the auditing dataset are publicly known during model training, which makes the model certification harder by allowing the modeler can adaptive training their model on the audit data. We are also using this approach to improve the reliability and robustness of fairness audits, which is seen as a promising direction for fairness certification.

B. Zero Knowledge Proof

Zero knowledge proofs were introduced by Goldwasser [30] and generic constructions based on probabilistically checkable proofs were proposed in the seminal works of Kilian [31] and Micali [32]. In recent years there has been significant progress in efficient ZKP protocols and systems. A radically different approach in zero-knowledge proof, categorized by their underlying techniques and assumptions, there are pairing-based schemes [24], [25], [27], discrete- log-based schemes [33], interactive-proof [34], [35], interactive oracle proofs (IOP) [26], [36], and so on. They provide different trade-offs between prover runtime, proof size and verifier runtime and so on. Please refer to [37] for more details on the performance and comparisons of different ZKP schemes. Zero knowledge proof has been widely used in blockchains and cryptocurrencies to achieve privacy [38] and scalability. More recently, it also found new applications in zero-knowledge machine learning [39], [40], zero-knowledge middlebox [41], and so on.

III. Preliminaries

In this section, we introduce the fairness notions in machine learning and the cryptographic primitives used in our framework.

Notions. We use λ to denote the security parameter. Let $[n]$ denote the set $\{0, 1, \ldots, n-1\}$; a vector be denoted by a boldface letter, e.g., $\mathbf{x}$. And $x \leftarrow \mathcal{X}$ denote that x is sampled from a distribution $\mathcal{X}$.

A. Fairness Notions in Machine Learning

There is plenty of fairness definitions [42] such as group fairness, causal discrimination, and counterfactual fairness. In this study, we mainly focus on statistical fairness definitions that require protected data, such as demographic parity (or statistical parity) [43], equalized odds [5], equal of opportunity [5], and disparate impact [44]. Demographic parity means that both protected and unprotected groups have an equal probability of being assigned to the positive predicted class. Equalized odds enforces both equal bias and equal accuracy in all demographics. Equal of opportunity is a relaxation of equalized odds, which only focus the positive predication outcome. Disparate impact implies that the decision outcomes disproportionately benefit or hurt members of certain sensitive attribute value groups.

Let M be a trained machine learning model for a classification task. Suppose possible inputs $\mathcal{X}$, sensitive or protected attribute $\mathcal{G}$ (relevant for fairness, e.g., ethnic or sex), the true class label y and the predication $\hat{y} = M(x, g)$, where $x \in \mathcal{X}, g \in \mathcal{G}$. And we use tuple $(M, \mathcal{X}, \mathcal{Y}, \mathcal{G})$ represent the audit sample D. Consider $g = 0$ designates the unprotected group and $g = 1$ designates the protected group. We recall these fairness definition below.

- Demographic parity (DP): $P(\hat{y} = 1|g = 0) = P(\hat{y} = 1|g = 1)$
- Equalized odds (EO): $P(\hat{y} = 1|y, g = 0) = P(\hat{y} = 1|y, g = 1), \forall y \in \mathcal{Y}$
- Equal opportunity: $P(\hat{y} = 1|y = 1, g = 0) = P(\hat{y} = 1|y = 1, g = 1)$
- Disparate impact (DI): $P(\hat{y} = 1|g) \neq P(\hat{y} = 1), \forall g \in \mathcal{G}$

The fairness of the ML model is assessed by means of the empirical fairness gap, as described in [19]. The fairness notion must be expressed in the formulation in order to audit ML models. Without loss of generality, we can consider the demographic parity:

$$l_{g,y}(M) = \mathop{\mathbb{E}}_{(x,g',y')} [\mathbb{I}\{M(x) = y\}|g' = g]$$

where $\mathbb{I}$ is an indicator function. Then define the estimated group risk as $l_{g,y}(M,T) = \frac{1}{m}\sum_{i=1}^{m_g} \mathbb{I}\{M(x_i) = y_i \wedge g_i = g\}$, where $T = \{(x_1, g_1, y_1), \ldots, (x_m, g_m, y_m)\}$ is independent sample set and m_g is the number of samples in T from group g. Define the fairness gap of ML model as

$$\max_{g_0,g_1 \in \mathcal{G}, y \in \mathcal{Y}} |l_{g_0,y}(M) - l_{g_1,y}(M)|$$

Also define empirical fairness gap (EFG) using the empirical approximation as follows:

$$EFG = \max_{g_0,g_1 \in \mathcal{G}, y \in \mathcal{Y}} |l_{g_0,y}(M,T) - l_{g_1,y}(M,T)|$$

We call model M (ϵ, δ)-fair on $(\mathcal{G}, \mathcal{T})$ with respect to a fairness measurement if:

$$\Pr\left[\max_{g_0,g_1 \in \mathcal{G}, y \in \mathcal{Y}} |l_{g_0,y}(M) - l_{g_1,y}(M)| > \epsilon\right] \leq \delta$$

The EFG can naturally extend to the other fairness notion and yield the corresponding (ϵ, δ)-fairness definitions.

B. Cryptography Primitives

Bilinear Groups. A bilinear group is given by a description $GK = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ such that

- $\mathbb{G}_1$, $\mathbb{G}_2$ are cyclic groups of prime order p and generators are $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$
- Bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, that is, $e(g^a, h^b) = e(g, h)^{ab}$, where $a, b \in \mathbb{Z}_p$
- $e(g, h)$ generates $\mathbb{G}_T$

Commitment Scheme. A commitment scheme allows a committee to commit a secret value and later open the commitment and reveal the value to the verifier. We recall the commitment scheme definition.

Definition 1: A commitment scheme is a tuple of algorithms $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerifyCommit})$ that works as follows.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ takes as input the security parameter λ and outputs a commitment key ck.
- $\mathsf{Commit}(\mathsf{ck}, m) \to (c, o)$ takes as input the commitment key ck and a secret value m, and output a commitment c_m and an opening o.
- $\mathsf{VerCom}(\mathsf{ck}, c_m, m, o) \to b$ takes as input a commitment c_m, a value m and an opening o, and output accept ($b = 1$) or reject ($b = 0$).

The commitment scheme is required to be both binding and hiding. In the study, we will be using Perdersen-like commitment scheme [27] which is statistically hiding and computationally binding under suit assumptions.

zkSNARKs. Zero knowledge proof enables a *prover* to prove to a *verifier* the result y of a computation $\mathcal{C}$ satisfying $y = C(x, w)$, where x is public input and w is secret witness of prover. The popular zero knowledge proof notions used in practice are zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs, for short) [23]–[25]. Here we recall the definition of zkSNARKs as follows:

Definition 2: A zk-SNARK for a relation R is a tuple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ as follows:

- $\mathsf{Setup}(1^\lambda, R) \to \mathsf{CRS} = ((ek, vk), \tau)$: The setup algorithm takes a relation $R \in R_\lambda$ and security parameter as input, and returns a common reference string CRS and a simulation trapdoor τ.
- $\mathsf{Prove}(ek, x, w) \to \pi$ The prove algorithm takes a evaluation key ek from CRS, and $(x, w) \in R$ as inputs, and generates a proof π.
- $\mathsf{Verify}(vk, x, \pi) \to b \in \{0, 1\}$ The verify algorithm takes a verification key vk from CRS, public input x, and proof π, and outputs 0(reject) or 1 (accept).
- $\mathsf{Sim}(\mathsf{CRS}, \tau, R)$ The Sim algorithm takes a CRS, a simulation trapdoor td, and a relation R as input, and returns a proof π.

A zkSNARKs scheme should satisfy the following properties:

- Completeness: For any pair $(x, w) \in R$, the verifier always accepts the corresponding proof.
- Knowledge Soundness: it holds if the prover must know a witness and such knowledge can be efficiently extracted from the prover by using a knowledge extractor.
- Zero knowledge: An argument is zero-knowledge if it does not leak any information other than the truth of the statement. There are exist a simulator without secrets can generate valid proofs.
- Succinctness: The size of a proof is $|\pi| \leq poly(k)polylog(|x| + |w|)$

Commit-and-prove SNARK. Commit-and-prove SNARK is a SNARK(cp-SNARK, for short) [27] that can prove knowledge of (x, w) such that $R(x, w) = 1$ holds w.r.t. a witness $w = (u, w)$ and u opens a commitment c_u as follow:

Definition 3: We denote a cp-SNARK as a triple of algorithm $\mathsf{CP} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerProof})$.

- $\mathsf{KeyGen}(ck, R) \rightarrow \mathsf{CRS} = (ek, vk)$ generates the common reference string(CRS).
- $\mathsf{Prove}(ek, x, (c_j)_{j\in[l]}, (u_j)_{j\in[l]}, (o_j)_{j\in[l]}, w) \rightarrow \pi$ outputs the proof of correct commitment.
- $\mathsf{VerProof}(vk, x, (c_j)_{j\in[l]}) \rightarrow b \in \{0, 1\}$ reject or accept the proofs.

The above definition has *perfect completeness*, *computational knowledge soundness* and *zero knowledge* in the random oracle model. Please refer to [27] for more details on the formal definition.

IV. Problem Statement

A. System Model

As shown in Fig. 1, our framework involves four entities: *server*, *regulators*, *client*, and *bulletin board*. With the involvement of the entities, we consider such a scenario problem: the server in possession of a trained model seeks to convince any later-coming client that the model satisfies a set of fairness metrics typically defined by a group of specialist regulators, while not revealing the model parameters. We note that we consider multiple specialist regulators who hold different fairness metrics dependent on the policy, law regulation, and environment of their domains, so as to ensure an all-around fairness assessment of a trained model. To address the above scenario problem, our framework contains three phases, including the query phase, the auditing phase, and the verification phase, with the following basic workflow:

- The sever commits to the model that will be evaluated, and meanwhile, the regulators commits to their test data that are used for evaluating model fairness.
- Any one regulator can send the test data to the server, and the model is evaluated on the data, thereby obtaining the corresponding evaluation result. Also, the commitment on the evaluation result and the proof regarding evaluation correctness are submitted to the bulletin board. Here refers to the query phase.
- A regulator can audit the model fairness with the evaluation result he obtains, according to the fairness metric he holds. As a result, the regulator submits the auditing result and a proof on correct auditing to the bulletin board. It refers to the auditing phase.
- Any later-coming client who questions the model fairness is able to browse the fairness auditing results and assert the truth with the proofs from the bulletin board. Here refers to the verification phase.

Remarks. We remark that the server provides a black-box query inference towards the regulators, without exposing the model parameters.

B. Threat Assumptions

We consider either the server or the regulators have the motivation to cheat the client with respect to the model fairness in our scenario problem. We now clarify our concrete threat assumptions on the involved entities.

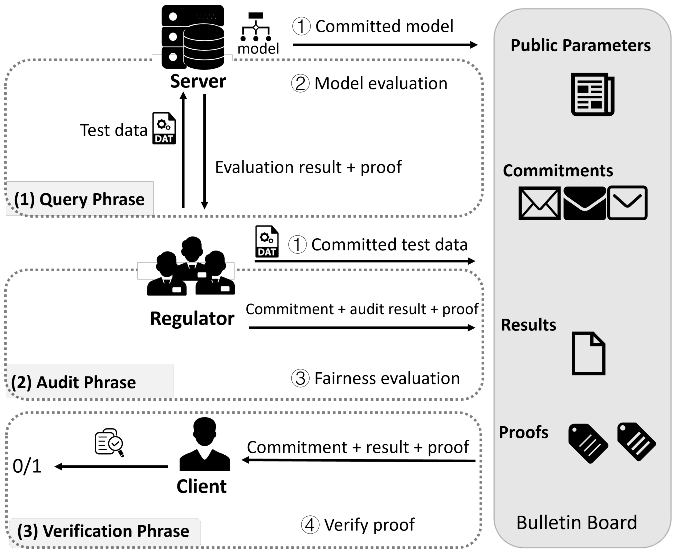


Fig. 1. System overview

Server: The server is considered to be *malicious*, which may arbitrarily deviate from the evaluation of model fairness. Specifically, 1) it may give incorrect model predication on test data, such as random values or predication not on the committed model or test data; 2) it also may use an unfair model to interact with the regulator, trying to trick the regulator. We note the server has access to the regulator's test dataset in plaintext in our scenario. First, given that the source and amount of data is limited, there is a tendency to audit model fairness on publicly available datasets [19] to obtain richer test data. Second, in scenarios where privacy is required for the test dataset, since each of the regulator's test data only contains sensitive attributes and an identifier as stated in work [14], this represents a lesser privacy risk.

Regulator: We consider regulator as *malicious*. The regulator might give wrong fairness audit results, such as auditing an unfair model as fair or vice versa, due to conflict of interest or just machine failure. The regulator also might use test datasets that differs from the previously committed data.

Client: We assume client is *honest*. The client can know the algorithms for model prediction and fairness evaluation, but does not have access to the model parameters. The commitments, audit result, and proofs are always available to client.

Bulletin Board: We assume that integrity and availability hold for bulletin board. The bulletin board can be instantiated by using a blockchain system, such as Bitcoin or Ethereum. We make the assumption like existing work [45]. We also assume the existence of a secure communication channel between any two entities.

Remarks. Note that we do not consider how to train a fair machine learning model and we do not discuss some machine learning attack, such as model extraction, model inversion, and evasion attacks [46] [47]. These studies are outside the scope of this work.

C. Security Goals

We aim to propose a framework for model fairness auditing in trustless setting, which departs from previous works. The framework establishes an evidence on the server-side model fairness, such that an off-line or later-coming client can be faithfully convinced of the truth respective to model fairness. To be specific, we should achieve the following security goals.

Trustless Verifiable Model Fairness: We require that the truth of model fairness can be efficiently verified by any one non-designated client. Concretely, a client of interest as a verifier can verify the public proofs posted by the server and the regulators, so as to determine if the server's model indeed reaches the fairness degree of the regulators over specific test data.

Model Privacy: We require that neither evaluation results, auditing results or proofs reveals the private information of the model against any one verifier.

Audit Integrity: We can provide the publicly verifiable audit integrity to convince that any later-coming client.

Accountability: We require that a verifier can account the misbehavior of the server or the regulator, if evaluation proof or audit proof cannot be verified. This property cannot be provided by MPC-based works.

V. Technical Challenge

Above scenario allows the server to know the regulator's test data for evaluating his model, and the regulators to obtain the corresponding evaluation results for auditing model fairness. When any a later-coming client gets the auditing results, we do not desire the client to learn any private information of the server's model, the regulator's test data and the evaluation results. We therefore need to achieve that any client (as non-designated verifier) can check the correctness of model evaluation and fairness auditing without private information of the model, test data and evaluation results.

Zero-knowledge proof technique can be used to address the above scenario problem without leaking private information. As Fig. 1 demonstrated, a regulator can firstly commit to test data (the server also commits to the model). The server then conducts the model evaluation with the test data, yielding the corresponding evaluation results. It also commits to the evaluation results, and generates a proof π_1 that the model is indeed evaluated over the test data, with the committed evaluation results as output. After that, the regulator executes the computations of fairness auditing, and generates a proof π_2 regarding the execution correctness. Lastly, a verifier checks the validity of both π_1 and π_2 without knowing the previously mentioned model, test data and evaluation results. Despite the easy-following technical roadmap, we encounter the following two challenges for efficiency:

A. Supporting Lightweight Verification

Our work adopts the state-of-the-art zkSNARK scheme constructed by Groth [25] (refer to Groth16) due to its short proof size (three group elements) and efficient verification. Despite the optimal performance of the scheme, the direct adoption without any modification cannot satisfy our scenario requirements. Specifically, to test model fairness and ensure the result reliability require sufficiently large test data, e.g., 6800 test inputs per regulator, as mentioned in [19]. Based on the scenario, the overall proof size will increase linearly with the amount of data, although the single proof of the Groth16 scheme is succinct. But a verifier is considered a thin device with restricted resources, and thus the result proof as described above easily becomes a computational and storage burden on the lightweight verifier. Furthermore, this situation becomes even worse when we want to verify the results of the fairness audit of multiple regulators with different test data and different fairness metrics. Therefore, the challenge is implementing more efficient verification when the model fairness is audited by large test data from multiple regulators.

B. Improving Proving Performance

Recall that the proof generation process based on an zk-SNARK scheme involves compiling a computation (e.g., model fairness auditing) into a circuit, such as arithmetic circuit and Boolean circuit, and then expressing the circuit with rank-1 constraint system (R1CS) used for generating proof. The involved arithmetic and Boolean operations generally determine the efficiency of the zkSNARK scheme used in practical. For example, the overhead of the Groth16 scheme is friendly for arithmetic operations in the finite field but unfriendly for Boolean operations (due to the radical blow-up in the circuit size required to compile Boolean operations into the arithmetic circuit). Such technical feature, however, is in conflict with the concrete computation of our scenario, since model fairness auditing involves many Boolean-efficient operations such as comparison. Our challenge thus is efficiently handling Boolean operations in proof generation.

VI. Concrete Design

We now present our framework that enables the fairness audit for an ML model while keep model confidentiality in an efficient publicly verifiable manner, enabling each one to assess model fairness individually and thus minimize trust dependency between server, regulator and client. It builds on Groth16 zk-SNARK scheme, Pedersen commitment and signature scheme. As mentioned above, our framework consists of three phase: *query phase*, *audit phase*, and *verification phase*. (1) In the query phase, the regulator query the server's model using the committed test dataset. Then the server evaluate model on test data and generate a proof for correct evaluation, and sends the evaluation results and the evaluation proof to the regulator. (2) The regulator verifies the evaluation proof and evaluate fairness metrics of model. And then the regulator generate auditing proof. And release auditing result and auditing proof to the bulletin board. (3) The client verifies the evaluation proof, audit proof and determines the fairness of the server's model.

A. Query Phase

1) Model evaluation.: In this phase, the main goal of regulator is to obtain the evaluation of the server's model on test data that can be used to audit the model's fairness. Firstly, the regulator collects sufficient test dataset $\mathcal{X} = \{x_0, \ldots, x_{n-1}\}$ (in a legally compliant way or directly extracted from public audit datasets [19]) and commits it to $\mathbf{cm}_x$ using the Pedersen commitment scheme. The server also commits its ML model parameter to $\mathbf{cm}_m$ using the same commitment scheme. Then both parties posting their commitments $\mathbf{cm}_x$ and $\mathbf{cm}_m$ to the Bulletin board respectively. Noted that the ML model structure is known to the verifier, we only protect the model parameter privacy, e.g. weight information. Secondly, the regulator transmits test data $\mathcal{X} = \{x_0, \dots, x_{n-1}\}$ to the server, then the server computes the evaluations of model on the test data $\hat{\mathcal{Y}} = \{y_0, \dots, y_{n-1}\}$ and generates the evaluation proof π_e. Moreover, the server commit the evaluation of model to $\mathbf{cm}_{\hat{y}}$ and publish it and the evaluation proof π_e to the Bulletin board, and the server sends the model evaluations $\hat{\mathcal{Y}}$ to the regulator.

2) Proof generation.: Our framework conducts the commit-and-prove paradigms [27] so that we can support zero-knowledge evaluation for both secret input and secret models in a straightforward way. Specially, in our scenario, we allow the server get the test data in plaintext, and the regulator obtains the evaluation results for audit the model. However, from the client's perspective, the privacy of the model, test data, and the evaluation result all is preserved.

The claim from [19] stated that an ML model M is ϵ-fair with confidence $1 - \delta$ if:

$$EFG < \epsilon \wedge \min_{g \in \mathcal{G}} \geq \frac{2}{(\epsilon - EFG)^2} \ln \frac{2|\mathcal{G}||\mathcal{Y}|}{\delta}$$

where m_g denotes the number of occurrences of g in T. Takes $EFG = 0.05, \epsilon = 0.1, \delta = 0.2$ and $|\mathcal{G}| = 100$ as example, we need sample number $m_g \approx 6800$. In this scenario, although the individual evaluation proof are small, thousands of test data make verifying multiple evaluation proofs expensive. As stated above, however, our goal is to keep the succinctness of proof due to we want to support a lightweight client who has limited memory and computation resources. There are two common techniques to keep succinctness of multiple proofs in literature, one is SNARK recursion [48] [49], and other is proof aggregation [50] [51]. The SNARK recursion can prove the proof is correct, and we can compress a sequence of proofs into one proof. Specially, we can aggregate proofs via recursive composition that create another SNARK for the circuit that contains n copies of the Groth16 verifier circuit [48]. But the SNARK recursion incur significant practical overhead due to we need to compiler the verify algorithm into a circuit, and this is the bottleneck of recursion SNARK efficiency. For example, computing a pairing on the BLS12-377 curve require ~ 15000 constraints [48].

3) Proof aggregation.: In our work, we adopt proof aggregation technique. Inspired by SnarkPack [50] [51], we resort to utilize special structure of proof to aggregate multiple proof. SnarkPack propose an approach to reduce the overhead in communication and verification time for verify multiple proofs without the need of further larger trusted setup ceremonies. The SnarkPack allows to aggregate n Groth16 zk-SNARKs proofs with $O(\log n)$ proof size and verifier time and can be constructed from two different existing ceremonies (e.g., the "power of tau" for Zcash [38] and Filecoin [52]).

We explain the aggregation protocol used in SnarkPack and how it can adapt to our CP-SNARK scenario below. First, we recall the verification process used in Groth16. A detailed description of Groth16 SNARK protocol can be found in [25]. The proof π in Groth16 consists of three group element $\pi = (A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$. For the verification algorithm, we need the verification key vk:

$$\mathsf{vk} := \left(g^{\alpha}, h^{\beta}, \left\{ S_j = g^{\frac{\beta_j(s) + \alpha w_j(s) + y_j(s)}{\gamma}} \right\}_{j=0}^{t}, h^{\gamma}, h^{\delta} \right)$$

The verifier need check that pairing equations is satisfy: $e(A, B) = e(g^{\alpha}, h^{\beta}) \cdot e(\sum_{j=0}^{t} S_j^{a_j}, h^{\gamma}) \cdot e(C, h^{\delta})$, where $[a]$ is public input.

The high level idea of Groth16 aggregation is straightforward: instead of checking that n different pairing equations are simultaneously satisfied, it is sufficient to prove that only one inner pairing product of a random linear combination of these equations defined by a verifier's random challenge $r \in \mathbb{Z}_p$ holds. The same idea is heavily exploited exploited in polynomial commitment, SNARK batch, SNARK recursive [48]. Specially, consider n proof $[\pi_i]_{i=0}^{n-1} = \{A_i, B_i, C_i\}_{i=0}^{n-1}$, the verifier need to check n equations of $e(A_i, B_i) = Y_i \cdot e(C_i, h^{\delta})$, where

$$Y_i = e(g^{\alpha}, h^{\beta})^n \cdot e(\prod_{i=0}^{n-1} S_i^{\sum_{j=0}^{n-1} a_{i,j} \cdot r^j}, h^{\gamma})$$

The aggregation will instead check a single randomized equation:

$$\prod_{i=0}^{n-1} e(A_i, B_i)^{r^i} = \prod_{i=0}^{n-1} Y_i^{r^i} \cdot e(\prod_{i=0}^{n-1} C_i^{r^i}, h^{\delta})$$

And we rewritten above equation as:

$$Z_{AB} = Y'_{prod} \cdot e(Z_C, h^{\delta})$$

where $Z_{AB} = \prod_{i=0}^{n-1} e(A_i, B_i)^{r^i}$, $Z_C = \prod_{i=0}^{n-1} C_i^{r^i}$ and $Y'_{prod} = \prod_{i=0}^{n-1} Y_i^{r^i}$. And then we will check that Z_{AB}, Z_C are consistent with the initial proof triples. Here we use two notions: the target inner pairing product (TIPP) and the multi-exponentiation inner product (MIPP) (detail can see [50]).

- TIPP: takes some committed vector $\mathbf{A} \in \mathbb{G}_1^n, \mathbf{B} \in \mathbb{G}_2^n$ and shows that $Z_{AB} = \prod_{i=0}^{n-1} e(A_i, B_i)$;
- MIPP: takes a committed vector $\mathbf{C} \in \mathbb{G}_1^n$ and a vector $\mathbf{r} \in \mathbb{Z}_p^n$ and shows that $Z_C = \prod_{i=0}^{n-1} C_i^{r^i}$

After that we can use TIPP and MIPP to generate the aggregated proof $\pi = (\pi_t, \pi_m, T_{AB}, U_{AB}, T_C, U_C, Z_{AB}, Z_C)$, where the last two elements are required to verify the Groth16 equation, the first two elements used to verify the TIPP and MIPP arguments, and other elements are required for the verifier derive randomness r in Fiat-Shamir transformation [53]. After verify TIPP and MIPP proof, the regulator use Z_{AB}, Z_C as the linear combination of the proofs. Then the regulator verify the Groth16 equation using the aggregated proof Z_{AB}, Z_C and decides whether to move to the next stage.

In the case of CP-SNARK, we need additional element D of the proof that contains a commitment to the data and to create a CPlink to link D to the external commitment. Also we need some additional element in CRS to create D and the CPlink. Nevertheless, the special structure of the proof does

TABLE I. CONFUSION MATRIX

	Actual-Positive	Actual-Negative
Evaluation-Positive	True Positive (TP) TPR=$\frac{\text{TP}}{\text{TP+FN}}$ PPV=$\frac{\text{TP}}{\text{TP+FP}}$	False Positive (FP) FPR=$\frac{\text{FP}}{\text{FP+TN}}$
Evaluation-Negative	False Negative (FN) FNR = $\frac{\text{FN}}{\text{TP+FN}}$	True Negative (TN) TNR=$\frac{\text{TN}}{\text{TN+FP}}$

not change. In order to verify the proof, we only need verify equation of the structure: $e(A, B) = Y \cdot e(C, h^{\delta})$. Thus, we can obtain the aggregated proof in the same way.

B. Auditing Phase

In this phase, multiple regulators aim to evaluate particular fairness metrics depending on the environment, policy, and industry to determine the fairness of the ML model.

1) Fairness audit: Following the common approach in [42], we can use a confusion matrix (see Table I) to compute statistical metrics of ML model, which is what most statistical measures of fairness rely on. The basic notion in confusion matrix as follow:

- True positive (TN): a case when the predicted and actual evaluation are both in the positive class.
- False positive (FP): a case predicted to be in the positive class when the actual outcome belongs to the negative class.
- False negative (FN): a case predicted to be in the negative class when the actual outcome belongs to the positive class.
- True negative (TN): a case when the predicted and actual evaluation are both in the negative class.

Based on these basic concepts, we calculate the fairness metrics. Specifically, the fairness notion we took in this work is as follows: Demographic parity (DP), Equalize odds (EO), Equal opportunity, and Disparate impact (DI). Use these formulation, following the basic paradigm in zk-SNARKs, we can express the whole fairness computation as arithmetic circuit and then generate proof.

2) Technical observation: As mentioned earlier, the challenge in generating proof is that we must handle many non-linear operations because we adopt group-based fairness notions, e.g., divides, comparison, sorting, and so on. To resolve the challenge, our framework uses the observation that for a prover to convince a verifier that it knows the output of some non-linear operation, the prover does not actually need to execute the non-linear operation in the circuit. Instead, the prover just needs to prove that the output of the non-linear operations is correct. For example, suppose the prover wants to prove $z = x/y$ to the verifier. In that case, the prover does not need to straight to compiler divide to a ciruit but simply provides a divided result z as an "advice" and then prove multiplication operation $x = z \times y$. Note that the multiplication operation only contain one multiplication gate, so it much more efficient than naive encoding the divide operation to a circuit (e.g., compute inverse use Fermat's little theorem, $a \cdot a^{p-2} = 1$). The observation is broadly used in literature [24], [39], [54], [55]. Next, we will show how to bring this idea into our framework to help model audit.

3) Handle non-linear operations: At a high level, we can split the computation in the audit phase into two-component: one is to evaluate fairness metrics, and the other is to compute the empirical fairness gap. First, we summarize the typical non-linear operation in the auditing process below and then design a protocol to prompt efficiency. The typical operations in statistical fairness evaluate as follow: 1) divides; 2) absolute value; 3) comparison; and 4) maximum or minimum value. To clarify the research methodology, we consider the following examples. First, the regulator needs to compute the basic metrics: TP, FP, TN, and FN. And then calculate fairness metrics like TPR and FRP. Finally, the regulator computes the EFG corresponding fairness notion, such as DP.

For division operations, we have solved as above. For comparison operation, e.g. $x \geq 0$, we ask the prover to provider the bit decomposition $(a_0, \ldots, a_k)$ of x as a witness (a_k denote sign bit, e.g. positive(1) or negative(0)). Then we can check that: 1) each a_i is binary: $a_i(a_i - 1) = 0, \forall i \in [k]$; 2) The correctness of bit-decomposition of x: $a_k(x - \sum_{i=0}^{k-1} a_i 2^i) + (1 - a_k)(x + \sum_{i=0}^{k-1} a_i 2^i) = 0$.

In calculating the EFG, we need to find the maximum metrics gap between different group. Naive computation is not feasible because it requires us to make a two-by-two difference between all the elements in the group and then select the maximum gap value. A more thoughtful way is to sort the array and then use the maximum value minus the minimum value to get EFG. Nevertheless, directly representing a sorting algorithm such as QuickSort as a circuit requires a comparison that contains $O(n \log n)$, which is expensive. Following the above observation, we asked the prover to provide some "advice" as a witness to improve efficiency, and we can combine absolute and maximum value into one relation. Concretely, assume that we have $(x_0, \ldots, x_{n-1})$ denote the fairness metrics on n test data and want to compute $\max |x_i - x_j|$ between different group. The prover is required to provide the maximum x_{max} and the minimum value x_{min} in list as an auxiliary witness. Then we can check that x_{max} is actually the maximum number as follows:

1) $x_{max} - x_j \geq 0 \ \forall j \in [n]$.
2) $\exists j \in [n]$ such that $x_{max} - x_j = 0$. This condition is equivalent to $x_{max} \cdot \prod_{j=0}^{n-1} (x_{max} - x_j) = 0$.

The first condition can be checked by bit-decomposing of $x_{max} - x_j$ and then checks are exactly the same as the comparison operation. Similarly, the check of x_{min} is exactly the same as x_{max} above.

Overall, comparing to straight compute maximize value above, the prover only additionally provides bit-decomposing of maximize and minimize value, and the protocol checks two additional bit decomposition. After all of computation done, the regulator publish all of the commitments, proofs, and audit result with associated signature on bullet board.

TABLE II. EVALUATION PROOF TIME OF FLR MODEL

Dataset	preprocess time(s)	prover time(s)	verifier time(s)
German	2.139	0.736	0.0159
Bank	110.853	34.9684	0.0293
Adult	82.272	26.921	0.0403

C. Verification Phase

Depending on the application area, regulator is selected to verify the correctness of the audit results. After get the audit result from y, the client request committed values for the ML models cm_m, test data cm_x, and evaluation results cm_y. The client then request evaluation proof π_e and auditing proof π_a from the bulletin board.

Upon reception of all of message, the client verifies the signature using the public key of the selected regulators and server for the authenticity of commitment, proof and results. If satisfied, client then verifies the evaluation proof π_e and audit proof π_a using the verification key and commitment. If all these verification pass, then the client will be convinced by the regulator's auditing results and thus determinate the fairness of the model.

If the client verify failure then one can identify the misbehaving party and take penalize it, in the form of reputation evaluation, incentives, etc. Then the party responsible for the misleading behavior is deterred and restart a new auditing process. After all checks are successful, the client can determine the correctness of the ML model auditing results.

VII. EVALUATION

We implemented our fairness audit framework and we present the experiment result in this section.

A. Setup

We have implemented the fairness audit framework in C++ using the libsnark [56] library. We run all of the experiments on 4-core Intel i7-5600k (2.6 GHZ, 8 physical cores) and 48 GB of RAM with Ubuntu18.04. Note that we run our experiments on Docker container. Our current implementation is only use a single CPU core. We report the time in seconds and take the average of 10 runs per experiment as the result.

We used three real-world datasets from various domains: German credit dataset (German), Bank marketing dataset (Bank), and Adult income dataset (Adult) [57]. The datasets vary in size and disparity of minority groups and as such some can be used to create fair or unfair models. In this experiment, we train fair logistic regression (FLR)[1]. Note that due to zk-SNARK systems only support group elements, we use a generic 8-bit unsigned quantization technique to transform float into integers.

[1]The implementation of fair machine learning model is based on the repository https://github.com/mbilalzafar/fair-classification.

B. Performance of Model Query

In this section, we report the performance of the query phase of the model. First, we use FLR model to measure the time to generate evaluation proofs for all three datasets. Since matrix multiplication is involved in all three models, a representative FLR is selected to measure the performance. Table II shows the results of evaluation proof in model query phase. The FLR model needs matrix-vector multiplication between the weight vector and input matrix of test data, where the sizes of input matrix are 1000×20 and 45550×20, and 48880×15. As a result, the input test data and the number of samples affect the circuit constraints and thus the performance.

After that we measure the overhead incurred by aggregating the proofs. Due to the different sizes of the three datasets to ensure the fairness of the model, we set different fairness gaps and confidence levels just like in [19]. We performed our test on the mentioned datasets with $\delta = 0.05$ and $\epsilon = 0.1$. Our evaluation result can see in Fig. 2. The results show that aggregated proofs of thousands of proofs are relatively small in size and can be verified in a few seconds. Although the proofs take more time, this is not impractical. In practice, aggregation operations can be performed offline without affecting the online use of the model after deployment.

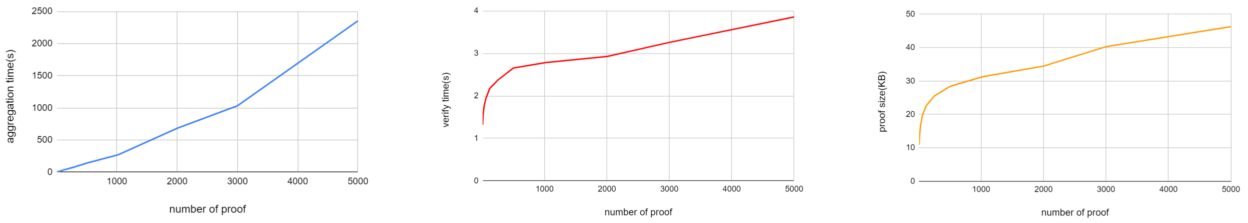
C. Performance of Fairness Audit

After query phase, the regulator calculates fairness metrics based on the evaluation results from the server. Thus, we measure computation times of fairness metrics in three datasets. Naturally, we split the data into a training and test subset. The sizes of the three datasets are 200, 9100 and 9000, respectively. In Bank dataset, we use the marital status feature as binary sensitive attribute and income for labels. We considered gender as a binary sensitive variable in Adult and German. Table III shows the evaluation results of fairness audit. The experimental results prove the usefulness of our framework, which takes only a few milliseconds to verify the correctness of the fairness audit results, making it easy for anyone to ensure that they are protected from discrimination. In addition, both preprocessing and prove times are within acceptable bounds.

VIII. DISCUSSION AND FUTURE WORK

One problem with the Groth16 system used in the framework instantiation, despite its state-of-the-art proof scale, is that it requires a trusted third party to generate the CRS used to construct the proofs. While it is possible to use MPC to generate reliable CRS [58], a transparent zkp system [28] could be used instead to avoid trusted party, which is left for future exploration. We expect that better ZK systems will emerge to replace the ZKP schemes we use and thus improve the efficiency of proposed framework. In order to achieve optimal ZK systems in all aspects, such as proof size, proofing time, etc., one promising direction is proof composition [37], [59]. Also, we can naturally extend our work to support confidential model prediction and model accuracy assessment [39], [40].

For future work, we want to explore further protecting the privacy of test data on the server. Especially since there are no publicly available test data and the user data is susceptible in some scenarios. A promising direction is the use of verifiable



(a) prover time (b) verifier time (c) proof size

Fig. 2. Performance of Aggregation

TABLE III. Fairness Audit Performance of FLR Model on Three Datasets

Dataset	preprocess time(s)				prover time(s)				verifier time(s)			
	DP	EO	EOP	DI	DP	EO	EOP	DI	DP	EO	EOP	DI
German	8.17	8.62	9.84	8.29	1.64	2.35	1.95	2.25	0.0062	0.0059	0.0061	0.0063
Bank	87.13	86.42	86.01	88.23	25.55	23.23	24.66	25.28	0.0052	0.0061	0.0063	0.0052
Adult	81.19	80.58	81.56	82.48	23.23	24.96	23.86	23.75	0.0067	0.0068	0.0069	0.0065

encryption techniques [60] to ensure integrity and confidentiality. Finally, it is also interesting to explore other fairness definitions beyond group-based as a research direction.

IX. Conclusion

This work proposes a framework to prompt publicly fair audits for a machine learning model. Unlike previous work, our construction only assumes the third-party collection test data and does not rely on the third party. We minimize the trust between the server, the regulator, and the client. Also, our framework can support multiple regulators to provide more strength and border fairness without additional trust assumptions. Our experimental evaluation confirms that our framework is practical for fairness auditing ML models with real datasets.

Acknowledgment

The authors would like to thank Dr. Jiasi Weng for valuable and helpful discussions on this work. Wuzheng Tan was supported by National Natural Science Foundation of China (Grant Nos. 62272199).

References

[1] S. M. D. A. C. Jayatilake and G. U. Ganegoda, “Involvement of Machine Learning Tools in Healthcare Decision Making,” *Journal of Healthcare Engineering*, vol. 2021, p. e6679512, 2021.

[2] T. Tulabandhula and C. Rudin, “On combining machine learning with decision making,” *Machine Learning*, vol. 97, no. 1, pp. 33–64, 2014.

[3] A. E. W. Johnson, M. M. Ghassemi, S. Nemati, K. E. Niehaus, D. A. Clifton, and G. D. Clifford, “Machine Learning and Decision Support in Critical Care,” *Proceedings of the IEEE*, vol. 104, no. 2, pp. 444–466, 2016.

[4] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, “Fairness-Aware Classifier with Prejudice Remover Regularizer,” in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Springer, 2012, pp. 35–50.

[5] M. Hardt, E. Price, E. Price, and N. Srebro, “Equality of Opportunity in Supervised Learning,” in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[6] G. Pleiss, M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger, “On Fairness and Calibration,” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[7] J.-G. Lee, Y. Roh, H. Song, and S. E. Whang, “Machine Learning Robustness, Fairness, and their Convergence,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD ’21. Association for Computing Machinery, 2021, pp. 4046–4047.

[8] H. Zhang, X. Chu, A. Asudeh, and S. B. Navathe, “OmniFair: A Declarative System for Model-Agnostic Group Fairness in Machine Learning,” in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD ’21. Association for Computing Machinery, 2021, pp. 2076–2088.

[9] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilović, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, and Y. Zhang, “AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias,” *IBM Journal of Research and Development*, vol. 63, no. 4/5, pp. 4:1–4:15, 2019.

[10] B. d’Alessandro, C. O’Neil, and T. LaGatta, “Conscientious Classification: A Data Scientist’s Guide to Discrimination-Aware Classification,” *Big Data*, vol. 5, no. 2, pp. 120–134, 2017.

[11] B. H. Zhang, B. Lemoine, and M. Mitchell, “Mitigating Unwanted Biases with Adversarial Learning,” in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES ’18. Association for Computing Machinery, 2018, pp. 335–340.

[12] R. Berk, H. Heidari, S. Jabbari, M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth, “A Convex Framework for Fair Regression,” 2017.

[13] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings,” in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.

[14] M. Veale and R. Binns, “Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data,” *Big Data & Society*, vol. 4, no. 2, p. 2053951717743530, 2017.

[15] N. L. Martinez, M. A. Bertran, A. Papadaki, M. Rodrigues, and G. Sapiro, “Blind Pareto Fairness and Subgroup Robustness,” in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021, pp. 7492–7501.

[16] P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. Chi, “Fairness without Demographics through Adversarially Reweighted Learning,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 728–740.

[17] S. Park, S. Kim, and Y.-s. Lim, "Fairness Audit of Machine Learning Models with Confidential Computing," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. Association for Computing Machinery, 2022, pp. 3488–3499.

[18] N. Kilbertus, A. Gascon, M. Kusner, M. Veale, K. Gummadi, and A. Weller, "Blind Justice: Fairness with Encrypted Sensitive Attributes," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 2630–2639.

[19] S. Segal, Y. Adi, B. Pinkas, C. Baum, C. Ganesh, and J. Keshet, "Fairness in the Eyes of the Data: Certifying Machine-Learning Models," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '21. Association for Computing Machinery, 2021, pp. 926–935.

[20] S. Pentyala, D. Melanson, M. De Cock, and G. Farnadi, "PrivFair: A Library for Privacy-Preserving Fairness Auditing," 2022.

[21] T. Cloosters, M. Rodler, and L. Davi, "TEEREX: Discovery and Exploitation of Memory Corruption Vulnerabilities in SGX Enclaves," in *29th USENIX Security Symposium*, 2020.

[22] T. Cloosters, J. Willbold, T. Holz, and L. Davi, "SGXFuzz: Efficiently Synthesizing Nested Structures for SGX Enclave Fuzzing," in *USENIX Security*, 2022.

[23] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. Association for Computing Machinery, 2012, pp. 326–349.

[24] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly Practical Verifiable Computation," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 238–252.

[25] J. Groth, "On the Size of Pairing-Based Non-interactive Arguments," in *Advances in Cryptology – EUROCRYPT 2016*, ser. Lecture Notes in Computer Science. Springer, 2016, pp. 305–326.

[26] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge," 2019.

[27] M. Campanelli, D. Fiore, and A. Querol, "LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. Association for Computing Machinery, 2019, pp. 2075–2092.

[28] S. Setty, "Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup," in *Advances in Cryptology – CRYPTO 2020*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2020, pp. 704–737.

[29] J. Bootle, A. Chiesa, Y. Hu, and M. Orrú, "Gemini: Elastic SNARKs for Diverse Environments," in *Advances in Cryptology – EUROCRYPT 2022*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2022, pp. 427–457.

[30] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing - STOC '85*. ACM Press, 1985, pp. 291–304.

[31] J. Kilian, "A note on efficient zero-knowledge proofs and arguments (extended abstract)," in *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '92. Association for Computing Machinery, 1992, pp. 723–732.

[32] S. Micali, "CS proofs," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 436–453.

[33] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 315–334.

[34] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation," in *Advances in Cryptology – CRYPTO 2019*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 733–764.

[35] J. Zhang, T. Liu, W. Wang, Y. Zhang, D. Song, X. Xie, and Y. Zhang, "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. Association for Computing Machinery, 2021, pp. 159–177.

[36] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable Zero Knowledge with No Trusted Setup," in *Advances in Cryptology – CRYPTO 2019*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 701–732.

[37] A. Golovnev, J. Lee, S. Setty, J. Thaler, and R. S. Wahby, "Brakedown: Linear-time and post-quantum SNARKs for R1CS," 2021.

[38] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized Anonymous Payments from Bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[39] T. Liu, X. Xie, and Y. Zhang, "zkCNN: Zero Knowledge Proofs for Convolutional Neural Network Predictions and Accuracy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. Association for Computing Machinery, 2021, pp. 2968–2985.

[40] J. Weng, J. Weng, G. Tang, A. Yang, M. Li, and J.-N. Liu, "pvCNN: Privacy-Preserving and Verifiable Convolutional Neural Network Testing," 2022.

[41] P. Grubbs, A. Arun, Y. Zhang, J. Bonneau, and M. Walfish, "{Zero-Knowledge} Middleboxes," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4255–4272.

[42] S. Verma and J. Rubin, "Fairness Definitions Explained," in *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*, 2018, pp. 1–7.

[43] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. Association for Computing Machinery, 2012, pp. 214–226.

[44] A. Chouldechova, "Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments," *Big Data*, vol. 5, no. 2, pp. 153–163, 2017.

[45] S. Kanjalkar, Y. Zhang, S. Gandlur, and A. Miller, "Publicly Auditable MPC-as-a-Service with succinct verification and universal setup," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2021, pp. 386–411.

[46] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18.

[47] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 253–261.

[48] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu, "ZEXE: Enabling Decentralized Private Computation," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 947–964.

[49] N. Tyagi, B. Fisch, A. Zitek, J. Bonneau, and S. Tessaro, "VeRSA: Verifiable Registries with Efficient Client Audits from RSA Authenticated Dictionaries," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2022, pp. 2793–2807.

[50] B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely, "Proofs for Inner Pairing Products and Applications," in *Advances in Cryptology – ASIACRYPT 2021*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2021, pp. 65–97.

[51] N. Gailly, M. Maller, and A. Nitulescu, "SnarkPack: Practical SNARK Aggregation," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2022, pp. 203–229.

[52] "Filecoin: A decentralized market for storage," https://filecoin.io.

[53] A. Fiat and A. Shamir, "How To Prove Yourself: Practical Solutions to Identification and Signature Problems," in *Advances in Cryptology — CRYPTO' 86*, ser. Lecture Notes in Computer Science. Springer, 1987, pp. 186–194.

[54] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou, "vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 863–880.

[55] S. Angel, A. J. Blumberg, E. Ioannidis, and J. Woods, "Efficient Representation of Numerical Optimization Problems for {SNARKs}," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4273–4290.

[56] "libsnark: a c++ library for zksnark proofs," https://github.com/sciprlab/libsnark.

[57] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[58] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza, "Secure Sampling of Public Parameters for Succinct Zero Knowledge Proofs," in *2015 IEEE Symposium on Security and Privacy*, pp. 287–304.

[59] T. Xie, Y. Zhang, and D. Song, "Orion: Zero Knowledge Proof with Linear Prover Time," in *Advances in Cryptology – CRYPTO 2022*, ser. Lecture Notes in Computer Science. Springer Nature Switzerland, 2022, pp. 299–328.

[60] D. Fiore, A. Nitulescu, and D. Pointcheval, "Boosting Verifiable Computation on Encrypted Data," in *Public-Key Cryptography – PKC 2020*, ser. Lecture Notes in Computer Science. Springer International Publishing, pp. 124–154.