Fairness as a Service (FaaS)

Toreini, Ehsan; Mehrnezhad, Maryam; van Moorsel, Aad

DOI:
10.1007/s10207-023-00774-z

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Toreini, E, Mehrnezhad, M & van Moorsel, A 2024, 'Fairness as a Service (FaaS): verifiable and privacy-preserving fairness auditing of machine learning systems', *International Journal of Information Security*, vol. 23, no. 2, pp. 981-997. https://doi.org/10.1007/s10207-023-00774-z

International Journal of Information Security (2024) 23:981–997
https://doi.org/10.1007/s10207-023-00774-z

REGULAR CONTRIBUTION

Fairness as a Service (FaaS): verifiable and privacy-preserving fairness auditing of machine learning systems

Ehsan Toreini[1] · Maryam Mehrnezhad[2] · Aad van Moorsel[3]

Published online: 7 November 2023

Abstract
Providing trust in machine learning (ML) systems and their fairness is a socio-technical challenge, and while the use of ML continues to rise, there is lack of adequate processes and governance practices to assure their fairness. In this paper, we propose FaaS, a novel privacy-preserving, end-to-end verifiable solution, that audits the algorithmic fairness of ML systems. FaaS offers several features, which are absent from previous designs. The FAAS protocol is model-agnostic and independent of specific fairness metrics and can be utilised as a service by multiple stakeholders. FAAS uses zero knowledge proofs to assure the well-formedness of the cryptograms and provenance in the steps of the protocol. We implement a proof of concept of the FaaS architecture and protocol using off-the-shelf hardware, software, and datasets and run experiments to demonstrate its practical feasibility and to analyse its performance and scalability. Our experiments confirm that our proposed protocol is scalable to large-scale auditing scenarios (e.g. over 1000 participants) and secure against various attack vectors.

Keywords Machine learning · Fairness · Trust · Zero knowledge proofs · Auditing

1 Introduction

Many of our ordinary and daily decisions are now made or assisted by machines. While the accuracy and efficiency of such systems matured, the social implications and aspects of these decisions are still unattested. Until recently, the issues of trust and trustworthiness were systematically ignored by system designers and downgraded to the lack of accuracy [3]. However, it has become clear in recent years that the trust of end users requires more presenting them with a highly accurate algorithm [2]. There is no shortage of examples that have diminished trust in algorithms because of unfair discrimination of certain groups [7, 41]. This includes human resource decision-making tools used by large companies, which turn out to discriminate against women [40]. There also are seminal examples studied widely within the academic community, such as the unfair decisions related to recidivism in different ethnic groups [28]. In the UK, the algorithm to determine the A-levels substitute scores under COVID-19 was widely found to be unfair across demographics [31]. Rise of large linguistic models (LLM) such as Chat-GPT 3.0 and GPT 4.0, or generative adversarial networks (GAN) resurfaced the necessity of auditing and regulating ML models in the public domain.

Demonstrating the fairness of algorithms is critical to the continued proliferation and acceptance (and trustworthiness) of algorithmic decision-making in general [32] and AI-based systems in particular [48]. Data scientists have attempted to formalise the notion of fairness and provided various metrics for measuring the level of discrimination in the outcome of ML systems. Many different metrics have been proposed including individual fairness and group fairness. Evidently, various expressions for fairness cannot be satisfied together [48]. Regardless of the fairness metric itself, to trust the ML systems one needs to place trust in the entity in charge of the computations of such a fairness metric, and, ideally, one is able to verify these computations and possibly appeal against the outcome. In other words, it is crucial to be able to audit the correctness of the fairness-enforcing components in an ML architecture. In current approaches, the ML owner (e.g. an AI

✉ Ehsan Toreini
e.toreini@surrey.ac.uk

Maryam Mehrnezhad
maryam.mehrnezhad@rhul.ac.uk

Aad van Moorsel
a.vanmoorsel@bham.ac.uk

[1] University of Surrey, Guildford, UK

[2] Royal Holloway University of London, Egham, UK

[3] University of Birmingham, Birmingham, UK

company) lists some specific notion(s) of fairness and claims their product complies with such notion(s), e.g. [12, 22, 27]. However, there is no easy way to audit such claims by any independent third-party, such as experts, social activists and end users. There has been research on auditability and transparency of ML models, e.g. [37], however, with no specific focus on fairness.

Interestingly, concerns about bias often intertwine with concerns about privacy of personal information. In general, a fair ML solution balances the output of the algorithm based on one or a combination of protected attributes such as gender, ethnicity, sexual orientation, economic status, etc. (attributes that are collectively referred to as sensitive attributes" in the literature [35]). Thus, the notion of fairness is inherently related to *privacy-sensitive attributes*, which should be protected and anonymised, along the lines imposed by GDPR [10]. In case there is a mechanism for auditing and verifying a model, the data owners want to be assured that none of the original data (typically sensitive and personal) is leaked. Similarly, the system that runs the algorithms (ML system) may have a valid interest in maintaining the secrecy of the model. These concerns should be considered in the design of a fairness auditing solution, following the privacy-by-design principles in the final product [16].

Based on the current research in the literature, our research questions are:

- *RQ1: How can we audit a model for fairness without revealing the model's internal configurations and datasets? (black-box auditing).*
- *RQ2: How can we compute the fairness without having access to sensitive information of the users (secure and privacy-preserving computation)?*
- *RQ3: How can anyone verify that the computation of the fairness is done correctly without trusting the auditor (universal verifiablility)?*

To answer the above questions, this paper provides a cryptographic solution to offer accountability to the computations of fair performances. We propose *Fairness as a Service (FaaS)*, as an independent fairness auditing service which performs verifiable and privacy-preserving black-box auditing of the models. FaaS facilitates end-to-end verifiability for computation of fairness to stakeholders via enabling everyone to audit the fairness of the ML algorithms by providing cryptographic proofs on each step of the computation without relying on any trusted third party. Our proposed auditing process follows a decentralised framework formed by a number of individuals (and independent) auditors, which we call auditors in the rest of the paper (Fig. 1). They individually audit the ML system and then send the encrypted response of the ML system (based on their data) to a designated service for computation (RQ1). Next, the computation server aggregates the cipher responses from all auditors (with no knowledge of the sensitive data from auditors) and computes the fairness performance of the ML system (RQ2). Moreover, the computation process is publicly verifiable as the encrypted information submitted by auditors together with the calculated fairness are posted publicly on a publicly accessible data structure. Any party can redo the computation of the fairness metric and verify the result is correct (RQ3).

Contributions: In our approach, we do not interfere and restrict the training of the model or rely on expensive high-end TEEs. Our design choices will make FaaS more cost-effective, while maintaining the same level of security guarantees such as privacy-preserving, verifiability and resilience against malicious actors. Also, the universal verifiability in FaaS (as other similar research does not provide) will enable everyone to actively engage in the process and prevent malicious activities. Summarising, our contributions are:

- We propose FaaS, a novel privacy-preserving, end-to-end verifiable architecture, in order to collectively audit the algorithmic fairness of ML systems. FaaS is model-agnostic (independent of the ML model) and takes a holistic approach towards auditing for group fairness metric.
- We design the FaaS protocol to ensure a privacy-preserving and universally verifiable solution. We use zero knowledge proofs (ZKPs) to assure the well-formedness of the cryptograms and the steps of the protocol.
- We implement a proof-of-concept of the FaaS architecture and protocol using off-the-shelf hardware, software, and datasets and run experiments to demonstrate the practical feasibility of FaaS. We perform experiments to analyse the performance and scalability of the system.

FaaS proposes a fundamentally different approach for secure black-box fairness auditing. The majority of the previous work is focused on a collaborative approach where the model is cryptographically entangled to a fairness definition during the training process [23, 26] or based on a designated hardware module for a secure computation environment (Trusted Execution Environments (TEE), e.g. intel SGX) with cryptographic commitments to secure execution of the audit [39].

Abbreviations: The list of the important abbreviations and what they stand for are presented for the reference throughout the manuscript. These terms are explained in text at length, when necessary. AI: *Artificial Intelligence*, ML: *Machine Learning*, ZKP: *Zero Knowledge Proof*, PA: *Protected Attributes*, DP: *Demographic Parity*, EOd: *Equalised Odds*, EOp: *Equality of Opportunity*, FCS: *Fairness Computation Service*.

2 Background and related work

One of the benefits of auditing ML-based products relates to trust. Trust and trustworthiness (in socio-technical terms) are complicated matters. Toreini et. al [48] proposed a framework for trustworthiness technologies in AI solutions based on existing social frameworks on trust (i.e. demonstration of Ability, Benevolence and Integrity, aka ABI and ABI+ frameworks) and technological trustworthiness [46]. They comprehensively reviewed the policy documents on regulating AI and the existing technical literature and derived any ML-based solution that needs to demonstrate fairness, explainability, auditability, and safety and security to establish social trust. Also, there are other proposals in the literature aiming to promote trustworthy machine learning notions [8]

The existing research in fair ML normally assumes the computation of the fairness metric to be done locally by the ML system, with full access to the data, including the private attributes [11, 12, 22]. However, there is a lack of verifiability and independence in these approaches, which will not necessarily lead to trustworthiness. To increase trust in the ML products, the providers might make the trained model self-explaining (a.k.a. transparent or explainable). There is also the transparent-by-design approach [5, 19, 50]. While this approach has its benefits, it is both model-specific and scenario-specific
[37]; thus, it cannot be generalised. There is also no trusted authority to verify such claims and explanations. Moreover, in reality, the trained model, datasets and feature extraction mechanisms are company assets. Once exposed, it can make them vulnerable to the competitors. Another approach to provide accountability to the fairness implementation comes through the black-box auditing, also known as ad hoc [19, 30, 38]. In this way, the model is trained and audited for different purposes [1]. This solution is similar to tax auditing and financial ledgers where accountants verify and ensure these calculations are legitimate. However, unlike the well-established body of certifications and qualifications for accountants in tax auditing and financial ledgers, there does not exist any processes and resources for fairness computation in AI and ML.

The concept of a service that calculates fairness has been proposed before, e.g. in [49]. The authors introduced an architecture to delegate the computation of fairness to a trusted third party that acts as a guarantor of its algorithmic fairness. In this model, the fairness service is trusted both by the ML system and the other stakeholders (e.g. users and activists). In particular, the ML system must trust the service to maintain the privacy of data and secrecy of its model, whilst revealing to the trusted third party the algorithm outcome, sensitive input data and even inner parameters of the model. This is a big assumption to trust that the third party would not misuse the information, and hence, the leakage of data and model information is not a threat.

To address these limitations, Kilbertdus et al. [26] proposed a system known as blind justice', which utilises multi-party computation protocols to enforce fairness into the ML model. Their proposal considers three groups of participants: User (data owner), Model (ML model owner) and the Regulator (that enforces a fairness metric). These three groups collaborate with each in order to train a fair ML model using a federated learning approach [51]. The outcome is a fair model that is trained with the participation of these three groups in a privacy-preserving way. They only provide a limited degree of verifiability in which the trained model is cryptographically certified after training, and each of the participants can make sure if the algorithm has not been modified. It should be noted that since they operate in the training stage of the ML pipeline, their approach is highly dependent on the implementation details of the ML model itself. Jagielski et al. [24] proposed a differential privacy approach in order to train a fair model. Similarly, Hu et al. [23] used a distributed approach to fair learning with only demographic information. Segal et al. [43] used similar cryptographic primitives but took a more holistic approach towards the computation and verification of fairness. They proposed a data-centric approach in which the verifier challenges a trained model via an encrypted and digitally certified dataset using Merkle tree and other cryptographic primitives. Furthermore, the regulator will certify the model is fair based on the data received from the clients and a set of dataset provided to the model. Their approach does not provide universal verifiability as the regulator is the only party involved in the computation of fairness. More recently, Park et al. [39] proposed a trusted execution environment (TEE) for the secure computation of fairness. Their proposal requires special hardware components, which are cryptographically secure and provide enough guarantees and verification for the correct execution of the code. Finally, Shamsabadi et al. [45] proposed a fully-verifiable protocol to audit decision trees for the integration of fairness in the learning process and beyond. They provided protocols to audit the models using their ZKP schemes.

The previous research generally has integrated fairness into their ML algorithms; therefore, such algorithms should be redesigned to use another fairness metric set. As it can be seen in Table 1, FaaS is the only work, which is independent of the ML model and fairness metric with universal verifiability and hence can be used as a service.

3 Preliminaries

In this section, we present the definitions required for our protocol and the FaaS Architecture.

Table 1 Features of FaaS and comparison with other privacy-oriented fair ML proposals (support: full: ✓, partial: ✙, none: ✗), the columns indicate the necessary qualities highlighted in this paper

Work Work	Universal verifiability	Independent of Fairness metric	Independent of ML model	User privacy	Model confidentiality	Off-the-shelf hardware
Veal and Binns [49]	✗	✗	✗	✗	✗	✓
Kilbertus et al. [26]	✙	✗	✗	✓	✓	✓
Jagielski et al. [24]	✗	✗	✗	✓	✗	✓
Hu et al. [23]	✗	✗	✗	✓	✗	✓
Segal et al. [43]	✙	✓	✓	✓	✓	✓
Park et al. [39]	✙	✓	✓	✓	✓	✗
Shamsabadi et al. [45]	✓	✗	✗	✓	✓	✗
FaaS (this paper)	✓	✓	✓	✓	✓	✓

3.1 Fairness metrics

Designing a fair algorithm requires being able to measure and assess the fairness. Researchers have worked on formalising fairness for a long time. Narayanan [36] lists at least 21 different fairness definitions in the literature, and this number is growing, e.g. [11, 12]. Fairness is typically expressed as discrimination in relation to data features. These features for which discrimination may happen are known as *Protected Attributes* (PAs) or sensitive attributes. These usually include, but are not limited to, ethnicity, gender, age, scholarity, nationality, religion and socio-economic elements.

The goal of training a model is usually tuned to reach the lowest possible error on unseen data. In the context of training a *fair ML model*, we extend this goal to maximise an specific notion of fairness while achieving the lowest possible error rate. In general, we assume that the fair model for binary classification problem is defined as:

Let Φ be the possible input, Ω be a finite set of groups that are essential in the definitions of fairness (PAs) and Υ be a finite set of labels. We define $\Phi \times \Omega \times \Upsilon$ in the probability space Λ with unknown distribution of Δ. A fair model is defined as $M_f(x)$ for classification where input $x \in \Phi$ with the ground truth denoted as $Y \in \Upsilon$. Ideally, the output $M_f(x)$ will be $\hat{Y} \in \Upsilon$ where for each datapoint x, $M_f(x)$ achieves: $min \left\| \hat{Y} - Y \right\|, \forall x \in \Phi$ and $max \left\| F(\hat{Y}, Y) \right\|, \forall Y, \hat{Y} \in \Upsilon$ where $F(Y, \hat{Y})$ is a fairness notion.

For the following definitions, let ϱ and A be an unknown datapoint where $(\varrho, A) \in (\Phi, \Omega)$. In this context, A denotes the PA and in what follows $A = 1$ expresses membership and $A = 0$ expresses no membership of a protected group. Model M is called fair on (Ω, Λ) where for $A \in \Omega$ and $A \in \{0, 1\}$, when it is trained to outcome an output $\hat{Y}$ on all unknown inputs $\forall \varrho \in \Phi$ as $Pr\left[max \left| M_f(\varrho) \underset{A=0}{} - M_f(\varrho) \underset{A=1}{} \right| \right] \leq \epsilon$, where ϵ is an threshold, usually set during the training process to optimise learning and fairness simultaneously [22].

For the baseline introduction, three fairness notions $F(\hat{Y}, Y)$ are defined as below:

(1) *Demographic Parity (DP)* A classifier satisfies DP when outcomes are equal across groups: $F_{DP} = \frac{Pr\left(\hat{Y}=1|A=0\right)}{Pr\left(\hat{Y}=1|A=1\right)}$

(2) *Equalised Odds (EOd)* A classifier satisfies EO if equality of outcomes happens across both groups and true labels: $F_{EOd} = \frac{Pr\left(\hat{Y}=1|A=0,Y=\gamma\right)}{Pr\left(\hat{Y}=1|A=1,Y=\gamma\right)}$ where $\gamma \in \{0, 1\}$.

(3) *Equality of Opportunity (EOp)* is similar to EO, but only requires equal outcomes across subgroups for *true positives*: $F_{EOp} = \frac{Pr\left(\hat{Y}=1|A=0,Y=1\right)}{Pr\left(\hat{Y}=1|A=1,Y=1\right)}$

Note that while we consider the above metrics for our protocol and proof-of-concept implementation in next sections, our core architecture is independent of metrics, and the metric set can be extended to other metrics defined beyond the notions described above.

3.2 Cryptographic primitives

FaaS protocol utilises variety of cryptographic primitives and concepts to achieve the objectives and research questions. In this section, we define the necessary primitives we deployed to implement FaaS protocol in more detail: zero knowledge proof and secure computation.

Zero knowledge proof (ZKP) schemes are predominantly used to provide correctness of the protocol operations and fairness calculations without compromising the privacy of the protocols actors and confidentiality of assets (such as dataset detail and more).

In the first stage, each individual auditor needs to demonstrate his knowledge of the exponent(s) without revealing it. We have used Schnorr's well-established signature [42]. Let H be a publicly agreed, secure hash function. In order to prove

the knowledge of the exponent (i.e. a secret ς_i for i-th individual auditor) for g^{ς_i}, one sends $(g^{\kappa}, r = \kappa - \varsigma_i \cdot h)$ where $\kappa \in \mathbb{Z}_q$ and $h = H(g, g^{\kappa}, g^{\varsigma_i}, i)$. This signature can be verified by anyone through checking whether g^{κ} and $g^{r} \cdot g^{\varsigma_i \cdot h}$ are equal.

For the auditing stage, we use the widely used 1-out-of-n interactive ZKP technique (famously known as CDS (Cramer, Damgard and Schoenmakers)) [13, 14], where $n = 8$ in our protocol (it could be extended to larger values for n, depending on the format of the auditing questions). For that, we should convert the terms of our protocol into the form of ElGamal encryptions. This can be done, in a universally verifiable fashion, by treating some elements of auditing computations as public key. We describe our protocols as interactive protocols with (semi) honest verifiers. One can obtain non-interactive arguments of knowledge in the Random Oracle model from them via Fiat–Shamir heuristic. Thus, by application of Fiat–Shamir heuristics [18], this ZKP can be converted into non-interactive, which makes the verification of proofs simpler [21].

*Secure computation*We let the protocol actors perform cryptographic operations without revealing their input, sensitive data and their tailored ML outcome. Secure computation can be imagined as a set of primitives that facilitates computation without the need for a trusted authority. This way, the *untrusted authority* receives the ciphered input from the participants (in encrypted form, i.e. cryptogram) and perform computations on them, producing an output without the need to decrypt them individually. Guarantees in these protocols can be given if at least one of the participants is acting honestly [43].

Secure computation schemes are mostly in two categories: *Homomorphic Encryption (HE)* and *Secure Multi-Party Computation (MPC)*. In this research, we opt-in to use HE methods as we assumed there will be no collaboration among the protocol actors (and between individual auditors), and the aggregation of their generated cryptograms would result in the computation of fairness metrics. The MPC methods usually are used in the use cases where there is a need for active engagement of actors in the mitigation of bias, e.g. where the model is in the earlier stages of ML pipeline, e.g. before deployment [26] or where the model engages in the auditing with prior knowledge [43]. In our approach, we deliberately discard the participation of ML system in the computation or audit process and assume that the ML system is semi-honest not aware of the auditing.

3.3 FaaS architecture

The overview of our FaaS Architecture is demonstrated in Fig. 1. FaaS has a black-box approach in auditing, meaning that it calculates fairness without leaking any personal data or model information (RQ1). The key elements of the

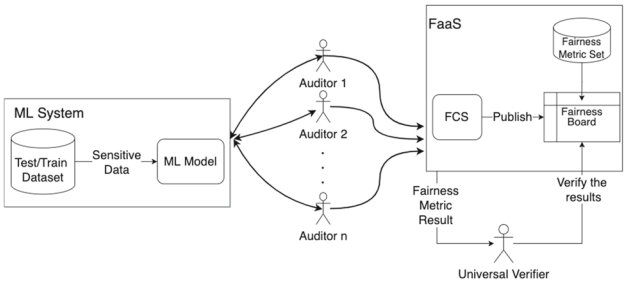

Fig. 1 FaaS architecture

sensitive data, required for the calculation of the fairness metrics, are encrypted (RQ2). The resulting cryptogram has a number of cryptographic properties to assure that the data cannot be changed and leaks no information. There are various ZKPs associated with the possible outcomes received from the auditors to guarantee that the data are valid (RQ3). In addition, the communication between the FaaS service and auditors is done through a secured channel in order to allow the two parties to identify each other and build a trust relation, although the exchange of encrypted data between protocol roles does not rely on a secure channel. Via the independent individual auditing, the model will be audited anonymously by a group of volunteers. The auditors will publish the individual cryptograms of the ML model results with proofs (without the need to reveal their sensitive attributes and the model outcome; achieving confidentiality and verifiability), and the individual audits will be published publicly. Anyone will be able to verify the published data to ensure the computations are correct.

3.4 Protocol actors

Our architecture includes four roles: (i) ML System: a system that owns the data and the ML algorithm, (ii) Individual Auditor: an individual that challenges the performance of the ML system (iii) Fairness Computation Service (FCS): a service that collects the cryptogram of the ML performance for each of the fairness auditors and computes the fair performance of the ML system, and (iv) Universal Verifier: anyone who has the technical expertise and motivation to verify the auditing process. These roles, their descriptions, output, trust assumptions are presented in Table 2.

Our architecture provides a privacy-preserving fairness auditing, which is independent of the ML systems and flexible with respect to the chosen fairness metrics. To establish this, we take the fairness measurement component out of the ML system and present it as a service, referred to as FCS. This architecture is presented as a service to all stakeholders, and parties can use it either with the aim to be audited or to challenge the fairness of ML system.

Table 2 FaaS roles and their features (FB: Fairness Board)

Protocol roles	Key feature	Output	Trust assumption	FB access
ML system	Owner of model and dataset	Trained and output accurate decision and commitment to abiding with one or more fairness metric(s)	Honest performance reporting (cross-validated with individual auditors), Incapable of recognising individual auditors from regular user	Read
FCS	Compute the fairness metric	Computation of Fairness, appending the cryptograms to the fairness board,maintenance of the fairness board	Secure storage secure key, selection of individual auditors	Read/write
Individual auditor	Auditing of ML	Cryptogram of model output for each auditor	Honest in reporting the ML output, secure storage of generated keys	Read
Universal verifier	Auditing of FCS	(Re)computation of fairness and verification of ZKPs published in Fairness Board	No assumption	Read

All the data required for fairness computations, along with the ZKPs are stored in a publicly accessible, append-only data structure named Fairness Board, hosted in FaaS server and with write-permission only restricted to FCS. Also, the FCS verifies the authenticity, correctness and integrity of data before publishing it. All steps in our protocols are stored in fairness board, Also, this data structure is accessible to everyone for universal, end-to-end verifiability.

3.5 Security and privacy assumptions

The design and implementation of the security of parties implementing the respective protocol roles (ML system, Individual Auditor, Fairness Computing Service, and Universal Verifier) (Figs. 1, 2) are independent of each other. The intercommunications that happen between the roles assume no trust between parties; thus, all their claims must be accompanied with validation proofs (for which we will use ZKP). The summary of our security and privacy assumptions for the protocol roles is shown in Table 2. In the context of this paper, we define privacy as described in [15].

FCS is hosted by an organisation, which is responsible to host the fairness auditing process. This organisation is also responsible for authenticating the auditors and verifying their identity to ensure each auditor only audits once. We assume the FCS is vulnerable to different attack arrays and not collusion-free. Thus, the data stored on the FCS must be encrypted, tamper-proof and verifiable at all stages. To provide such features, various cryptographic schemes are implemented by industrial standards (e.g. secure storage, tamper-resistant module for secure management of private keys, the sand-boxed cryptographic operations and deployment of secure execution environments such as secure enclave) [4]. The stored data on the FCS is end-to-end verifiable through ZKP protocols. Also, the publicly available data on FCS are fully encrypted and will not reveal any detail about the identity, demographic group and choices of the auditors. Moreover, the communication channel between the ML system, auditor and FCS is secure and encrypted. Finally, we assume that the cryptographic primitives and protocol have carefully been implemented (formally verified implementation).

In the design of the protocol, we opt to use an agnostic approach; the choice of the model in the ML system will not impact the operation of the protocol. The internal security of the ML system is beyond FaaS. The ML system needs to consider extra measures to protect its data and algorithms. We assume that the ML system does present the data and predictions honestly. This is a reasonable assumption since the incentives to perform *ethically* are in contrast to being dishonest when participating in fairness auditing process. However, the ML system will be independently audited; thus, the dishonest behaviour will be detected over time once the decentralised auditing results differ with the initial result beyond a pre-defined range tolerance. We assume that the ML system will not be able to differentiate auditors from ordinary users and hence cannot provide a tailored response to poison the process.

3.6 Limitations

The FaaS protocol has limitations in the current form. We acknowledge these limitations, and we will address them in future versions of the protocols. First, there is a lack of certification from the ML system, meaning that we assumed the ML model is trained fairly and the current protocol is designed to confirm the fairness of a fair model. The protocol does not provide guarantees and certifications for the ML model's fairness before its deployment though.

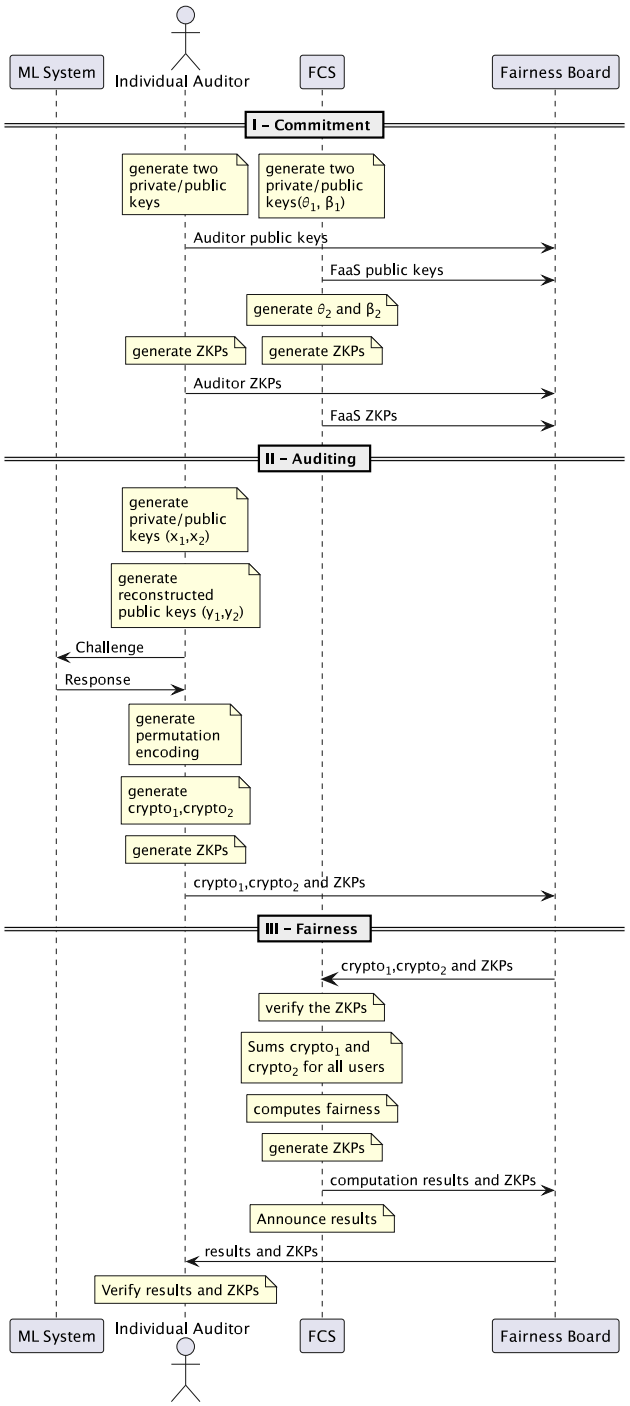

Fig. 2 FaaS protocol overview

The second problem is that the current permutation demography is limited to binary decision (e.g. True/false) and binary-sensitive attribute (e.g. Male/Female). The protocol is designed in a way that it can be extended to more complicated demographics; however, the computational complexities increase exponentially (as will be discussed later) consequently.

Third, is the limitations in the fairness notions. In the current definition, the fairness notions are limited to a category of definitions known as group fairness. In this category, the fairness definitions assume the datapoints in groups and enforced a balance outcome of ML model across them [17, 22]. For instance, the balance is maintained by enforcing equal true positive in groups of privileged and unprivileged users (e.g. gender) in hiring process.

The last limitation is the search space for the computation of fairness. With increase in the number of datapoints and individual auditors, the search space increases. We will propose some strategies to mitigate these issues in the design of our protocol.

4 Protocol description

In this section, we describe the details of our FaaS protocol. The detailed protocol sequence diagram is provided 2. The main security protocol sequence is between two actors, the individual auditor and FCS. Any universal verifier or ML system can turn to the FCS (which presents the fairness board), if they want to challenge the computations. The ML system is responsible for the implementation and execution of the ML algorithm. It has data as input and performs some prediction (depending on the use-case and purpose). We assume the ML system has already submitted their fairness metric. The Fairness Auditor challenges the ML system individually and generates the auditing cryptogram based on the received response and transmits the encrypted form of the response to the FCS with the relevant ZKPs. Then, the FCS evaluates the ML system's fair performance by first verifying the cryptograms and ZKPs and then computing a fairness metric. It also publishes the calculations on the Fairness Board. The FCS only has the right to append data (and the sufficient proofs) to the fairness board. In addition, the FaaS server verifies the authenticity, correctness and integrity of data before publishing it.

Our protocol sequence (Fig. 2) has three main stages: (i) Commitment, (ii) Auditing and (iii) Fairness Evaluations. We assume the protocol functions in multiplicative cyclic group setting (i.e. Digital Signature Algorithm (DSA)-like group [25]), but it can also function in additive cyclic groups (i.e. Elliptic Curve Digital Signature Algorithm (ECDSA)-like groups [25]). The auditors and FCS publicly agree on (p, q, g) before the start of the protocol. Let p and q be two large primes where $q|(p-1)$. In a multiplicative cyclic group $(\mathbb{Z}_p^*)$, G_q is a subgroup of prime order q and g is its generator. For simplicity, we assume the decision Diffie–Hellman (DDH) problem is out of scope [47]. The protection the private key pair depends on the security architecture of the system and protocol entities. We assume the private key is securely stored in standard practice (e.g. using

Table 3 Possible permutations of 3-bit representation of an entry in the original data

Membership of sensitive group	Actual label	Predicted label	Encoded permutation	Permutation #
No	0	0	000	#1
No	0	1	001	#2
No	1	0	010	#3
No	1	1	011	#4
Yes	0	0	100	#5
Yes	0	1	101	#6
Yes	1	0	110	#7
Yes	1	1	111	#8

the secure memory module on board or trusted platforms especially dedicated to secure encryption/decryption).

4.1 Phase I: commitment

In this phase, the auditors and FCS commit to each other and generate the foundations of the computations of fairness. First, the auditor and FCS agree on a set of values, which will be used as a token for the generation of auditing and fairness computation results. If the values that agreed in this stage get maliciously modified by anyone, it will make the whole computation of fairness invalidated. All the computations and commitment tokens will be provided with relevant ZKPs to guarantee the well-formedness and verifiability of the process. In this stage, the FCS will commit to each auditor separately, and the cryptograms generated at the end of this stage will be publicly stored in the fairness board for universal verifiability. Each auditor and FCS will store the private values at this stage securely and use the commitment for the purpose of generating relevant cryptograms required for computation of fairness by FCS in next stages.

Step 1: after initial agreements, the FCS randomly generates two private keys as $(k_1, k_2) \in \mathbb{Z}_p$ and computes the corresponding public keys for each of them as $(\mu_1 = g^{k_1}, \mu_2 = g^{k_2})$. The FCS publishes the public keys and the corresponding ZKPs on the fairness board for public verifiability (as $PW[k_1 : \mu_1]$ and $PW[k_2 : \mu_2]$). The ZKPs convince the auditors that the FCS owns the private keys for μ_1 and μ_2. Step 2: once FCS publishes μ_1 and μ_2 and ZKPs, each auditor (A_i, where $i \in 1, 2, \ldots, n$, n is the number of chosen auditors for the computation of the fairness) reads μ_1 and μ_2 and verifies the correctness and well-formedness of the ZKPs. Then, each auditor A_i generates two random private keypair $((a_i, b_i))$ using DSA or ECDSA and then computes their public keys as $\alpha_{1i} = g^{a_i}$ and $\beta_{1i} = g^{b_i}$. Like the FCS, each auditor publishes the public keys and the ZKPs $PW[a_i : \alpha_{1i}]$ and $PW[b_i : \beta_{1i}]$ to convince FCS and public verifiers that they know a_i and b_i. Step 3: after all n members of the auditor community generates and publish their public keys (α_{1i} and β_{1i}), the FCS will verify the well-formedness and correctness of the ZKPs recorded on the fairness board and start the last stage of the commitment phase. Step 4: finally the FCS generates another set of commitments $(\alpha_{2i}, \beta_{2i})$ for all the auditor A_i, where $i \in 1, \ldots, n$. FCS computes the commitments as: $\alpha_{2i} = \left(\frac{g}{(\alpha_{1i})^{\mu_1}}\right)^{\frac{1}{\mu_2}}$, $\beta_{2i} = \left(\frac{1}{(\beta_{1i})^{\mu_1}}\right)^{\frac{1}{\mu_2}}$

Then, the FCS will provide two sets of proofs to guarantee two conditions on the mathematical relationship between α_{1i}, α_{2i}, β_{1i} and β_{2i} as $PW[\alpha_{1i}, \alpha_{2i}, \mu_1, \mu_2]$ and $PW[\beta_{1i}, \beta_{2i}, \mu_1, \mu_2]$. The first ZKP will assure the $\alpha_{1i} \times \alpha_{2i} = g$ and the second will assure $\beta_{1i} \times \beta_{2i} = 1$ for all the αs and βs computed for all auditors A_i where $i \in 1, \ldots, n$ and n is the number of chosen auditors. At the end of this stage, the FCS and all auditors commit to each other and they have agreed on set of preparations for the auditing process to begin. Any malicious modifications to any parameters agreed on this stage cannot be undetected.

4.2 Phase II: auditing

In phase I, the auditors and FCS will commit to each other through a series of cryptographic schemes. In phase II, every auditor independently challenges the ML system and records the ML system's response; then, they will generate an encrypted form of their audit (referred to as *audit cryptogram* in the rest of this paper) and transmit it to FCS. When all other auditors finished the commitment phase (phase I), the auditor will be shown the auditing questionnaire.

Due to the nature of our protocol, this communication between ML system and the auditor should not be distinguishable from ordinary user's query from the ML's perspective. Once all the questions are answered, the auditor cryptogram is generated with the ZKPs and transmitted to the FCS. The auditor will receive the confirmation of the receipt of the auditing cryptogram and can verify it later in the fairness board. Once the auditor A_i receives the response from the ML system, it will start the generation of the *audit cryptogram* and the relevant ZKPs. This

process will have the following steps: Step (1): each auditor A_i will generate two random private keys as (s_{1i}, s_{2i}) and computes their corresponding public keys as $(g^{s_{1i}}, g^{s_{2i}})$. Then, A_i publishes the public keys and ZKPs for the private keys (as $PW[s_{1i} : g^{s_{1i}}]$ and $PW[s_{2i} : g^{s_{2i}}]$) in the fairness board. Step (2): Once all public keys are generated and published on the fairness board for all n auditors, each auditor A_i will compute two numbers $g^{r_{1i}}$ and $g^{r_{2i}}$, computed using $g^{r_{1i}} = \frac{\prod_{j=1}^{i-1} g^{s_{1j}}}{\prod_{j=i+1}^{n} g^{s_{1j}}}$, $g^{r_{2i}} = \frac{\prod_{j=1}^{i-1} g^{s_{2j}}}{\prod_{j=i+1}^{n} g^{s_{2j}}}$. We refer to these as *reconstructed public keys* as it is computed using a combination of public keys of all other auditors, except for the current one (i.e. A_i). These two elements $g^{r_{1i}}$ and $g^{r_{2i}}$ terms as public keys for the non-interactive CDS ZKP.

Step (3): At this step, the auditor will answer the questionnaire question based on their offline interaction with the ML system. Their response to the questionnaire will be used to generate the *permutation*, which is the combination of three parameters (which will only be known to the auditor and no one else): membership of a underprivileged demographic group, the response that they *should have* received from the ML system, the response that they *have* received from the ML system. This is a set of three binary parameters which forms into 8 possible permutations (as shown in Table 3). The auditors determine the form of decision that they *should* have received based on the subject of the problem they are auditing. For our use-case scenario, this can be done by sending list of eligibility criteria that are used to make a decision for mortgage to the auditors by the FaaS organisation. However, the auditors response to the three parameters are local and private, and only the auditor will have access to them. However, the auditor will generate the cryptogram of the audit permutation in the form of a pseudo-random sequence to the FCS for computation of the fairness of ML system. We will call this cryptogram the *audit cryptogram*.

Audit cryptograms: Each permutation is encoded into a $C_i = g^{p_i}$ which are computed based on the multi-option voting schemes introduced in [9] and applied in [20, 21]. p_i is computed based on the number n of auditors and m as the smallest integer such that $2^m > n$. For each of the eight permutations (as shown in Table 3), the p_i is computed as in Eq. 1.

$$p_i = \begin{cases} 2^0 & for\ permutation\ \#1 \\ 2^m & for\ permutation\ \#2 \\ \cdots & \cdots \\ 2^{7.m} & for\ permutation\ \#8 \end{cases} \tag{1}$$

The auditor A_i will generate C_i as the corresponding option for their audit permutation. Then, A_i will generate a random number in as γ_i and compute the corresponding public keys as $\Gamma_i = g^{\gamma_i}$. Then, it will compute the audit cryptograms using the following equation: $crypto_{1i} = g^{r_{1i} \cdot s_{1i}} \times (\alpha_{1i})^{C_i} \times (\beta_{1i})^{\gamma_i}$, $crypto_{2i} = g^{r_{2i} \cdot s_{2i}} \times (\alpha_{2i})^{C_i} \times (\beta_{2i})^{\gamma_i}$

Zero knowledge proofs: In addition to auditing cryptograms, the ML system also generates 1-out-of-8 ZKP for the audit permutation as $PW_i[crypto_{1i}, crypto_{2i} : s_{1i}, s_{2i}, g^{r_{1i}}, g^{r_{2i}}, \beta_{1i}, \beta_{2i}, \alpha_{1i}, \alpha_{2i}]$. This proof ensures the values presented as C_i in the auditing cryptogram indeed belong to g^{p_i} where $p_i \in \{2^0, 2^m, \cdots, 2^{7.m}\}$. The audit cryptogram contains a ZKP to guarantee it is one of the *valid* values for evaluating the fairness metric in next stages. In other words, the ZKP proves that the following statement Σ stands, where $\Sigma = ((g^{r_{1i} \cdot s_{1i}} (\alpha_{1i})(\beta_{1i})^{\gamma_i}) \wedge (g^{r_{2i} \cdot s_{2i}} (\alpha_{2i})(\beta_{2i})^{\gamma_i})) \vee ((g^{r_{1i} \cdot s_{1i}} (\alpha_{1i})^{2^{(m)}} (\beta_{1i})^{\gamma_i} \wedge (g^{r_{2i} \cdot s_{2i}} (\alpha_{2i})^{2^{(m)}} (\beta_{2i})^{\gamma_i})) \vee \cdots \vee ((g^{r_{1i} \cdot s_{1i}} (\alpha_{1i})^{2^{(7m)}} (\beta_{1i})^{\gamma_i}) \wedge (g^{r_{2i} \cdot s_{2i}} (\alpha_{2i})^{2^{(7m)}} (\beta_{2i})^{\gamma_i}))$

The auditor digitally signs the auditing cryptogram as $(crypto_{1i}, crypto_{2i}, \Gamma_i)$ along with ZKPs for $PW[s_{1i} : g^{s_{1i}}]$ and $PW[s_{2i} : g^{s_{2i}}]$ and Σ. Then, they will send it to the FCS for the computation of the fairness. The auditor also will publish these information on the fairness board for public verifiability. Finally, the audit cryptogram and relevant ZKPs will be digitally signed by the auditor and transmitted to FCS for the final stage. At the end of this stage, the FCS will have all the required information to compute the fairness metric.

4.3 Phase III: fairness evaluation

This phase starts when all the auditor successfully sent their permutation cryptogram and ZKPs to the FCS. The output of this process is the privacy-preserving and verifiable computation of fairness, which is a collective outcome of the audits done by the auditors. The FCS will announce the computations of the fairness metric so everyone else (i.e. the ML system, auditors, universal verifiers or members of the public) are able to re-do the computations and confirm whether the computations have been done in honesty. Also, the information for the computation of the fairness metric is publicly available in the fairness board with relevant ZKPs so any universal verifier can repeat the same steps for computation of the fairness and fully verify its correctness. In our proposed architecture, the FCS is not considered as trusted third party (TTP), and all the interactions are presented with sufficient verifiable proofs. All the required steps in this phase are executed in the FCS execution platform. The steps for this phase are as follows:

Step (1): the FCS receives the audit cryptograms from all auditors and verifies the ZKPs.

Step (2): In order to obtain the overall permutation numbers of the auditing cryptograms, the FCS sums the permutation cryptograms received from all n auditors (as $(crypto_{1i}, crypto_{2i})$, where $i \in [0, n]$) in the following

Table 4 The required permutations to compute the fairness metrics of an ML system, in the above formulas, n: total no. of auditors

Fairness component	Corresponding permutation #	Computation
$Pr(\hat{Y} \mid A = 0)$	#2, #4	$(\#2 + \#4)/n$
$Pr(\hat{Y} \mid A = 1)$	#6, #8	$(\#6 + \#8)/n$
$Pr(\hat{Y} \mid A = 0, y = 0)$	#2	$\#2/n$
$Pr(\hat{Y} \mid A = 1, y = 0)$	#6	$\#6/n$
$Pr(\hat{Y} \mid A = 0, y = 1)$	#4	$\#4/n$
$Pr(\hat{Y} \mid A = 1, y = 1)$	#8	$(\#8)/n$

manner:

$$
\begin{aligned}
fairness_j &= \prod_{i=1}^{n} (crypto_{ji}),\ j \in 1, 2 \\
&= \prod_{i=1}^{n} g^{r_{ji}.s_{ji}} (\alpha_{ji})^{C_i} (\beta_{ji})^{\gamma_i},\ j \in 1, 2 \\
&= g^{\sum_{i=1}^{n} r_{ji}.s_{ji}} . \prod_{i=1}^{n} (\alpha_{ji})^{C_i} (\beta_{ji})^{\gamma_i},\ j \in 1, 2
\end{aligned}
\tag{2}
$$

Then, it starts the process of computing the fairness metric. At this stage, the key point is the consideration of the effect (r_{1i}, r_{2i}) and (s_{1i}, s_{2i}) have on each other; known as Cancellation Formula" ([6, 20, 21]). Based on this formula, we have $\sum_{i=1}^{n} r_{ji}.s_{ji} = 0$, where $j \in 1, 2$. Considering the Cancellation Formula, we can conclude multiplication of all cryptograms into

$fairness_j = \prod_{i=1}^{n} (crypto_{ji}) = \prod_{i=1}^{n} (\alpha_{ji})^{C_i} (\beta_{ji})^{\gamma_i}$, where $j \in 1, 2$. The FCS will then compute $fairness = (fairness_1)^{(k_1)}.(fairness_2)^{(k_2)}$ and will publish this computation on the fairness board along with the ZKP for the computation of $fairness$ as $PW[fairness : fairness_1, fairness_2, \mu_1, \mu_2]$. All other verifiers will be able to verify the correctness and well-formedness of computation.

Given $fairness = (fairness_1)^{(k_1)}.(fairness_2)^{(k_2)}$, the overall fairness will be:

$$
\begin{aligned}
fairness &= \prod_{i=1}^{n} ((\alpha_{1i})^{C_i}.(\beta_{1i})^{\gamma_i})^{k_1}.((\alpha_{2i})^{C_i}.(\beta_{2i})^{\gamma_i})^{k_2} \\
&= \prod_{i=1}^{n} ((\alpha_{1i})^{k_1}.(\alpha_{2i})^{k_2})^{C_i}.((\beta_{1i})^{k_1}.(\beta_{2i})^{k_2})^{\gamma_i}
\end{aligned}
$$

From the preceding, we already know that $\alpha_{1i} \times \alpha_{2i} = g$ and $\beta_{1i} \times \beta_{2i} = 1$ for all the α's and β's computed for all auditors A_i where $i \in 1, \ldots, n$ and n is the number of chosen auditors. Hence, the final value for $fairness = \prod_{i=1}^{n} g^{C_i} = g^S$, where $S = \sum_{i=1}^{i=n} C_i$. Finally, a brute-force search is performed to get the sum of all individual audit permutations.

The result is the total sum of permutations (p#1 to p#8) as $\sum_i C_i = a.2^0 + b.2^m + c.2^{2m} + d.2^{3m} + e.2^{4m} + f.2^{5m} + g.2^{6m} + h.2^{7m}$ where a, b, c, d, e, f, g, h are the number of occurrence for each permutation, respectively (Table 3).

The total number of occurrence for the permutations ($fairness = \prod_{i=1}^{n} g^{C_i} = g^S$, where $S = \sum_{i=1}^{i=n} C_i$) determined by implementing a proper search algorithm. The search space for such combination depends on the number of auditors participating in the process (n: number of auditors for 8 permutations; $\binom{n+8-1}{8-1}$ [21]). In cases where the number of auditors is large (e.g. more than 100 auditors), the FaaS organisation can divide the auditors in smaller groups (e.g. 20 auditors in each group) and perform the FaaS protocol phases for each of them independently (in parallel). Then, the overall numbers for each permutation can be summed together to make a large-scale auditing of the ML system feasible. On the other hand, the universal verifier can follow the same steps to verify the fairness metric computations using the fairness auditing table that is publicly accessible via the fairness board. However, they do not require to perform the brute-force search. Instead, they will simply verify if the announced results (as S) equal the $\prod_{i=1}^{n} g^{C_i}$ by re-doing the multiplication of the audit cryptograms.

At the end of this stage, the FCS uses the acquired numbers to compute the fairness metric and releases the information publicly. The number of each permutation impacts the overall performance of the ML algorithm for each of the groups with protected attributes. Table 4 demonstrates the permutations and how it relates to the fairness metric of the ML system. The results will be published on the fairness board.

5 Implementation and evaluation

We implemented and evaluated our FaaS protocol. Our system has three parts: (i) the core (back-end) handling the computations and cryptographic operations, (ii) the auditing service authenticates the auditors and ML, receiving the commitment and cryptograms from the auditors and ML system, storing it on the fairness board and sending them to the back-end, (iii) the audit apps (client Python app) that the auditors and ML system use to generate the auditing cryptograms and ZKPs (two separate systems for each audit protocol).

Table 5 Performance results for a single auditing interaction (s), for laptop and cluster configuration, Bold (per step and totals) indicates the setting in which the Auditor runs on the laptop and the FCS on the server cluster

	Laptop mean			Server cluster		
		STD	Stage total	Mean	STD	Stage total
Stage I						
FCS generates random keypair (k_1, k_2)	0.254	0.018	**3.848** (6.393)	**0.074**	0.005	**0.784** (1.852)
FCS generates ZKP for μ_1	0.122	0.007		**0.038**	0.003	
FCS generates ZKP for μ_2	0.121	0.007		**0.037**	0.003	
Auditor verifies ZKP for μ_1	**0.385**	0.031		0.111	0.005	
Auditor verifies ZKP for μ_2	**0.383**	0.029		0.111	0.007	
Auditor generates keypair (a,b)	**0.254**	0.018		0.074	0.005	
Auditor generates ZKP for α_1	**0.127**	0.011		0.037	0.002	
Auditor generates ZKP for β_1	**0.127**	0.011		0.037	0.002	
FCS verifies ZKP for α_1	0.383	0.029		**0.111**	0.005	
FCS verifies ZKP for β_1	1.3E−06	3.0E−06		**2.9E−07**	1.8E−07	
FCS generates α_2	0.255	0.026		**0.074**	0.003	
FCS generates β_2	0.256	0.029		**0.074**	0.004	
FCS generates ZKP for α_2	0.513	0.043		**0.148**	0.008	
FCS generates ZKP for β_2	0.512	0.043		**0.148**	0.007	
Auditor verifies ZKP for α_2	**1.414**	0.091		0.407	0.022	
Auditor generates ZKP for β_2	**1.285**	0.091		0.369	0.017	
Stage II						
Auditor generates x_1, x_2	**0.257**	0.022	**15.163**	0.074	0.003	4.356
Auditor generates ZKP for x_1	**0.129**	0.014		0.037	0.002	
Auditor generates ZKP for x_2	**0.129**	0.013		0.037	0.002	
Auditor reconstructs public keys (y_1, y_2)	**0.046**	0.017		0.013	0.004	
Auditor generates audit cryptogram	**0.662**	0.047		0.190	0.008	
Auditor generates ZKP for Γ	**0.129**	0.010		0.037	0.002	
Auditor generates 1-out-of-8 ZKP	**13.811**	0.662		3.967	0.098	
Stage III						
FCS verifies ZKP for x_1	0.385	0.028	**0.383** (16.847)	**0.111**	0.005	**4.743** (4.855)
FCS verifies ZKP for x_2	0.385	0.029		**0.111**	0.005	
FCS verifies ZKP for Γ	0.387	0.032		**0.111**	0.005	
FCS verifies 1-out-of-8 ZKP	14.467	0.762		**4.176**	0.107	
FCS computes the overall fairness	0.316	0.105		**0.085**	0.006	
FCS generates ZKP for computations	0.522	0.029		**0.149**	0.003	
Auditor verifies ZKP for fairness	**0.383**	0.029		0.111	0.006	

5.1 Proof-of-concept implementation

The back-end is implemented in Python v3.7.1. The elliptic curve operations use the Python package *tinyec* and the conversion of Python classes to a JSON compatible format uses *JSONpickle*. Our experiments platform includes: (1) a MacBook pro laptop (CPU 2.7 GHz Quad-Core Intel Core i7 with 16 GB Memory running MacOS Ventura v.13.0.1), (2) a high-performance infrastructure: 56-core Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz with x86-64 architecture, with 256Gb RAM running CentOS Linux 7 (Core) for FCS and ML system—in a separate computational pipeline.

For this evaluation, we use a publicly available dataset from Medical Expenditure Panel Survey (MEPS) [29] that contains 15,830 data points about the healthcare utilisation of individuals. We developed a model (logistic regression) that determines whether a specific patient requires health services, such as additional care. This ML system assigns a score to each patient. If the score is above a preset threshold, then the patient requires extra health services. In the MEPS dataset, the protected attribute is race. A fair system provides such services fairly independent of the patient's race. Here, the privileged race group in this dataset is white ethnicity. We have used 50% of the dataset for training, 30%

for validation and the remaining 20% for testing. We set the number of cryptogram table samples to equal the size of test set ($N = 3166$). In this example, we include three attributes in the cryptogram to represent the binary values of A, Y and $\hat{Y}$ (Sect. 3.1), thus leading to 8 permutations for each data sample. As the design of FaaS is model-agnostic, one can replace our suggested system with other usecase domains as long as it remains in the 8 permutation response. (The extension of permutations will be discussed later.) The implementation details are available for researchers upon request.

5.2 Performance

We simulated 1000 auditors in our experimental settings. We report the average and standard deviations for the computations required for each auditor in Table 5. Comparing the cluster environment with the laptop, the laptop consistently takes about 3.5 times longer than the cluster to complete each step. For each calculation step, the standard deviation is low enough so that the width of a 99% confidence interval is less than 1% of the mean (i.e. we conclude that 99% of measurements will fall within this small interval around the mean). The exceptions to this are as follows. The very small values in phase I for FCS verify ZKP for β_1 exhibit larger standard deviation, but this may be caused by the small values and does anyway not impact overall results. More interesting is the step Auditor reconstructs public keys (r_1, r_2) in phase II, which has a confidence interval width of about 4% around the mean, both in the laptop and cluster experiments. Standard deviation of that particular computation is still reasonable on both platforms, albeit significantly larger than all other steps. In the laptop environment, the phase III step FCS computer the overall fairness has similarly high standard deviation, but not in the cluster environment. Altogether, on both platforms, 1000 samples are easily sufficient to obtain reliable results for the time taken at each computational step.

Table 5 shows the performance times. For stage I to complete takes 3.8 s on the laptop and 0.8 s on the cluster. Stage 2 concerns only the auditor, which takes 15.1 s on the laptop. The final stage is mostly carried out by the FCS, taking 4.7 s of computation time on the cluster, while the Auditor involvement on the laptop is limited to 0.38 s. In stage I, the most intensive operations belong to generation and verification of ZKPs. On the laptop, ZKP for μ_1 and μ_2 takes 120 ms with STD of 7 ms for generation and 385 ms with roughly 30 ms STD for verification of ZKPs, generation of ZKP for α_1 and β_1 took 120 ms with 10 ms STD, their verification also took 380 ms for α_1 and 0.13 ms for β_1. The same actions took 75 ms with 3 ms STD and 1.4 s with 90 ms STD for generation and verification of ZKP for α_2 and β_2, respectively.

In phase II, each auditor generates the audit cryptogram and the relevant ZKPs. The generation of private/public keypair (s_1, s_2) takes 25 ms with 2 ms STD. Once all auditors finished generation of the key pairs and published the public keys to the FB, each auditor will record the public keys and computes the reconstructed public keys (as r_1, r_2) in 4 ms with 1 ms STD. The generation of Γ and γ takes 12 ms with 1 ms STD on average. As expected, the most intensive step in this phase is the generation of ZKPs. The most complicated ZKP in FaaS protocol is the generation of 1-out-of-8 ZKP, which elapsed 13.8 s with 6 ms STD on average for each individual audit. The generation of other ZKPs took roughly 12 ms with 1 ms STD. Similar to phase I, the expected execution duration on cluster experimental set-up is roughly 4 times less than laptop setup. Note that generation of ZKP for 1-out-of-8 is an independent operation which can be executed in parallel for each auditor.

The verification of ZKPs in Phase III is a computationally expensive operation. The verification of ZKP for the ownership of the private keys took around 111 ms on average with standard deviation of 5 ms on laptop setting. The verification of 1-out-of-8 ZKP for each audit cryptogram roughly took 4 s on average with 100 ms standard deviation in the same execution setting. The generation and verification of the fairness computation ZKP also took 128 ms with 2 ms STD and 111 ms with 6 ms STD, respectively, in laptop. Same as before, the relevant duration on cluster configurations tool roughly 4 times faster than the laptop setting.

5.3 Scalability

For scalability experiments, we started with small-scale group (as of 5 simulated auditor) and we incrementally added 5 simulated auditor until we reached 100 individuals. Figure 3 shows the linear relation between the number of auditors and the elapsed time for phases I and III (3 attributes, 8 permutations). Stage II's performance is not dependent on the number of individuals as each auditor is supposed to perform the auditing separately. The scalability of the protocol depends on the computational power for collective work done by FCS in stages I and III, as it is responsible for performing the overall verification and generation received from each auditor. Based on our linear extrapolation analysis, the total elapsed time for finishing the tasks related to FCS for 1000 auditors will be 10 min for stage I and roughly 1.5 h in stage III on cluster setting, with most of the time elapsed on the verification of the ZKPs. As seen in Table 3, the majority of the computation at this stage is due to the verification of the ZKPs, while the actual computation of the Fairness is negligible in comparison with these figures.

In our experiments, we evaluated FaaS with 3 binary attributes, resulting in 3-bit encoding and 8 permutations. However, depending on the fairness metrics, the number of the required attributes would change. For instance, if we use the demographic disparity metric (Eq. 3.1), we only require access to the binary encoding of sensitivity group member-

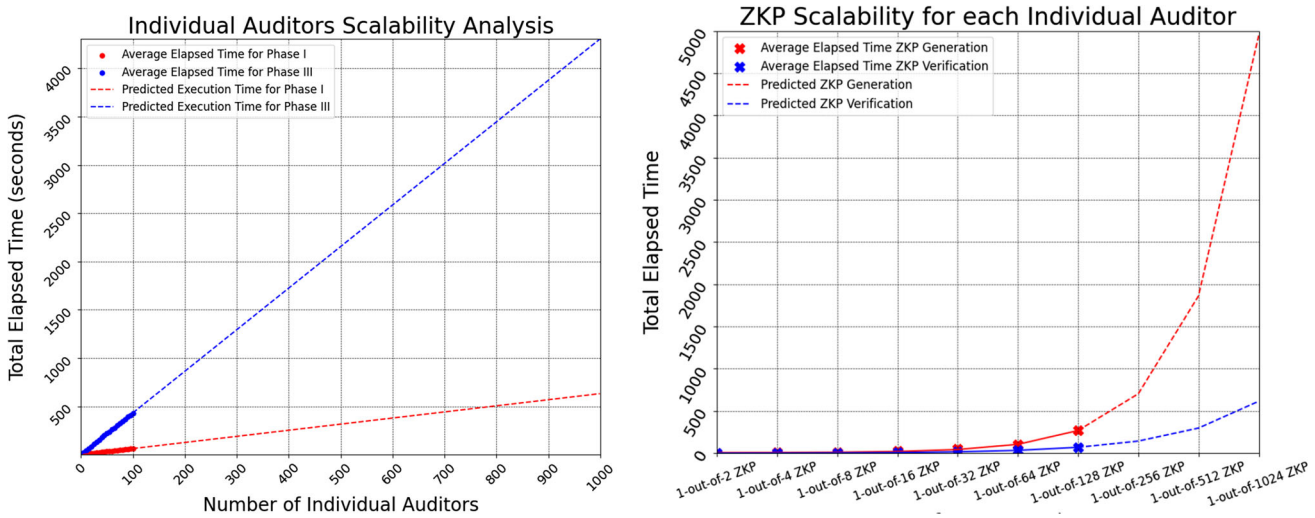

Fig. 3 Average elapsed time (in s) for (left) varying number of individual auditors, (right) varying number of permutations

ship and predicted label ($\hat{Y}$). Therefore, the encoding will be 2-bit encoding, resulting into 4 permutations. For more complicated cases such as the non-binary labels for the ML system outcome or the existence of more than one sensitive group for auditors, we might need to perform encoding with 3 or more bits.

We conducted another experiment to evaluate the impact of the number of attributes on the performance of our protocol. Here, we varied the number of attributes (bits), from 1 to 7 resulting in 2 to 128 permutations (i.e. ask auditors 7 binary questions). We recorded the performance of FaaS protocol 10 times for each permutation number. Figure 3 presents the average elapsed time for the generation of the cryptogram, fairness auditing and computation of fairness for a single auditor. As the increase in the permutation number is exponential, we performed an exponential regression model for our recorded performance and predicted FaaS performance in case one increases the number of permutations. The relevant elapsed time for generation (Phase II in laptop configurations) and verification (Phase III in cluster configuration) is also shown. In our experiment, the generation of a 1-out-of-128 ZKP took 269 s for each auditor (with 27 s STD), in contrast for around 69 s for the verification of the same ZKP (with 0.45s STD). Consequently, the verification of all 1-out-of 128 ZKP took roughly 20 h for $n = 1000$ auditors. Note that generation of ZKP for 1-out-of-128 is an independent operation which can be executed in parallel for each auditor; therefore, the overall elapsed time for such operation is not included here.

Figure 3 demonstrates the predicted number of options to 10 bits, resulting into 1024 predictions. In other words, the scalability of the protocol is determined particularly by Phase III, exhibiting an exponential growth. In our exponential regression analysis, the exponential growth for 10 bit permutation (as 1-out-of-1024 ZKP, which equals to asking auditors 10 binary questions) is predicted to execute in 2144 s for generation (in laptop configuration) and 596 s for verification (in cluster configurations), resulting into overall elapsed time of verification of ZKP for $n = 1000$ auditors to roughly 7.5 days.

6 Security and privacy analysis

First, we present set of the theoretical assumptions fundamental for our security analysis and proofs. Then, we discuss how FaaS protects anonymity, secrecy and integrity in computation of various fairness metrics. The FaaS protocol provides a publicly available FB as to facilitate universal verifiability. That is, all cryptograms are made available to the public and in particular to the individual auditors, who are able to verify that their audit request is represented in the fairness calculation. Of course, the FB is also available to any malicious party to perform different attacks, passive or active. Moreover, the attacker might target the individual auditor, FCS servers and the communication between them to manipulate the auditing process. We base our security analysis on the assumptions described in Sect. 3.5.

6.1 Assumptions and lemmas

In this section, we have set of fundamental assumptions prior to the security analysis. These assumptions will be used for the proofs presented later in security analysis. The theoretical assumptions are as follows [6, 20, 21]:

Assumption 1 Decisional Diffie–Hellman (DDH) Assumption. Given $g, g^a, g^b in G$ for uniformly chosen $a, b \in \mathbb{Z}$, it

is computationally hard to distinguish g^{ab} from a random number in G.

Assumption 2 Given g, g^a, g^b in G for uniformly chosen $a, b \in \mathbb{Z}$, it is computationally hard to distinguish g^{ab} from $g^{ab}.g^a$.

Assumption 3 Given g, g^a, g^b in G for uniformly chosen $a, b \in \mathbb{Z}$ and t in $\mathbb{Z}_q$, it is computationally hard to distinguish g^{ab} from $g^{ab}.g^t$.

The assumptions 2 and 3 are extensions of the DDH assumption as follows:

Lemma 1 *Assumption 2 implies Assumption 1.*

Proof Under the assumption set by the DDH, $(g, g^a, g^b, g^{ab}) \overset{c}{\approx} (g, g^a, g^b, R)$ where R is a random number in G. Thus, $(g, g^a, g^b, R) \overset{c}{\approx} (g, g^a, g^b, R \times g^a) \overset{c}{\approx} (g, g^a, g^b, g^{ab} \times g^a)$. Here, we extended the conclusions of Assumption 1 to 2. □

Lemma 2 *Assumption 3 implies Assumption 1.*

Proof Under the assumption set by the DDH, $(g, g^a, g^b, g^{ab}) \overset{c}{\approx} (g, g^a, g^b, R)$ where R is a random number in G. Thus, $(g, g^a, g^b, R) \overset{c}{\approx} (g, g^a, g^b, R \times g^t) \overset{c}{\approx} (g, g^a, g^b, g^{ab} \times g^a)$. Here, we extended the conclusions of Assumptions 1 to 3. □

In the rest of the proofs, we will refer to s_{1i} or s_{2i} as conclusive x_i and r_{i1} or r_{2i} as y_i for simplicity.

6.2 Audit secrecy

The data an individual Auditor submits to the FCS will be published on the FB, to allow people to verify all auditor cryptograms are taken into consideration, and the fairness calculation has been correct. This allows for malicious attempts to gain private information or to modify contents. In passive attacks an attacker may carry out any analysis of the data available from the FB to deduce useful information. In active attacks, attackers try to modify data, e.g. by exploiting a software bug in the FCS or poisoning the communication channel. We will show that neither passive nor active attacks can be successful in our scheme.

Proposition 1 (Data Leakage in passive attacks) *The attacker has access to the published cryptograms in the FB. Based on this information, under the DDH assumptions, attackers cannot distinguish any chosen audit cryptogram* $g^{r_i.s_i} \times (\alpha_{li})^{C_i} \times (\beta_{li})^{\gamma_i}$, *where* $l \in 1, 2$, *from random.*

Proof Attackers will know the following information from the FB: $g^{r_{li}.s_{li}}, \alpha_{li}, \beta_{li} (l \in 1, 2)$, a ZKP that proves the individual auditor system knows $s_{li} (l \in 1, 2)$ and a ZKP that proves $C_i = g^{p_i}$ where $p_i \in 2^0, 2^m, \cdots, 2^{7.m}$ and $m \in \mathbb{N}$ *and* $2^m > n$. Under the DDH assumptions, the attacker will not be able to distinguish $g^{s_{li}.r_{li}} (l \in 1, 2)$ from random (Assumption 2). Moreover, it will not be able to distinguish the combination $g^{s_{li}.r_{li}}.g^{p_i} (l \in 1, 2)$ (Assumption 3). These steps can be extended to the audit cryptogram recorded in the FB (as $g^{r_{li}.s_{li}} \times (\alpha_{li})^{C_i} \times (\beta_{li})^{\gamma_i}$, $p_i \in \{2^0, 2^m, \cdots, 2^{7.m}\}$, where $m \in \mathbb{N}$ *and* $2^m > n, l \in 1, 2$). Also, The first ZKP only reveals one bit of information: whether the individual auditor knows the discrete logarithm x_{li} of $g^{s_{li}} (l \in 1, 2)$. The second ZKP only reveals whether the message is well-formed (in other words, if it is actually the ElGamal encryption of the statement $p_i \in 2^0, 2^m, \cdots, 2^{7.m}$ where $m \in \mathbb{N}$ *and* $2^m > n$). Of course, this proof assumes the ZKP is constructed correctly [13]. The theoretical proofs for the security and correctness of the Schnorr's signature and Cramer et al.'s one-out-of-n ZKP are in [13, 42, 47] □

Proposition 2 (Stealthy modifications in active attacks) *The attacker with capability of malicious modification of the audit cryptogram cannot modify the cryptograms in the FB without being detected.*

Proof As before, the attackers will know the following information from the FB: $g^{r_{li}.s_{li}}, \alpha_{li}, \beta_{li} (l \in 1, 2)$, a ZKP that proves the individual auditor system knows $s_i (l \in 1, 2)$ and a ZKP that proves $p_i \in 2^0, 2^m, \cdots, 2^{7.m}$ where $m \in \mathbb{N}$ *and* $2^m > n$. Based on the proofs presented in Proposition 1 and Assumptions 1, 2 and 3, attackers cannot distinguish any chosen audit cryptogram $g^{r_{li}.s_{li}} \times (\alpha_{li})^{C_i} \times (\beta_{li})^{\gamma_i}$, where $l \in 1, 2$, from random. Thus, they will not be able to deduce useful data from merely eavesdropping the communication. However, they have the option to modify the data to deceive the FCS and manipulate the fairness computation. As the transmitted data are digitally signed by the individual auditor, any malicious modifications on the channel will result in rejection of the digital signature at the FCS side. Also, such modifications will cause into the rejection of ZKPs. As a whole, such modifications cannot be undetected and the attacker cannot deceive the computation of fairness into another one. □

Proposition 3 (Anonymity and privacy of the individual auditors) *The attacker cannot distinguish any specific individual auditor based on the FB data.*

Proof The above propositions (Proposition 1 and 2) prove the secrecy of the audit cryptogram, resulting into privacy preservation of the individual auditor as the cryptograms will not reveal the sensitive information of any specific individual auditor (according to Assumption 3). Such secrecy will also provide anonymity as the data recorded on FB will not provide detail of the individual auditor. Therefore, the attacker

is not able to link the data on FB to any specific individual auditor. The FB will contain audit cryptograms and ZKPs for the auditing of ML system; thus, it will not reveal anything other than the overall number of occurrence of each of the permutations as whole. The computation of the fairness metric is merely possible by overall summation of all the cryptograms. Hence, it preserves the anonymity of the individual auditor. □

6.3 Dispute freeness

Our proposed protocol satisfies dispute freeness in two ways. First, by using an authenticated channel between individual verifiers and the FCS, we ensure each individual auditor only audits the ML performance once. Secondly, the extensive implementation of ZKPs at various stages guarantees the well-formedness and correctness of the computations and the audit cryptogram, This, along with the universal verifiability of the announced computations ensures that the computations are not disputable.

6.4 Malicious actors

The collusion attack refers to a set of plans to plot against the correct execution of the auditing process. A hypothetical collusion can be done through attacks such as poisoning the audit cryptogram or trying to organise a large-scale coordination attempt by convincing the individual auditors to reveal specific parts of their audit cryptogram. In this attack, the attacker will not be able to tamper with the audit cryptogram (as shown in Propositions 3 and 1 and 2), instead it should directly influence the generation of the audit cryptograms. In the most serious case, it can convince them to reveal their private keys. In this situation, the attacker will be at the position to generate fake audit cryptograms.

Proposition 4 (Collusion attack) *Under the DDH assumptions, attackers with partial knowledge of the s_{1i} and s_{2i} (as defined in Sect. 4.1) cannot distinguish r_{1i} and r_{2i} (as defined in Eq. 4.2) from random.*

Proof As the attacker has partial knowledge, in the worst-case scenario, it knows all $s_{lj}, (j \neq k, l \in 1, 2)$ except for one pair $s_{lk}, l \in 1, 2)$. In this case, $s_{lk}, l \in 1, 2$ is uniformly distributed over $\mathbb{Z}_q$ and is unknown to the attacker. Since $r_{li}, l \in 1, 2$ is derived form all $s_{lp}, (p \neq i, l \in 1, 2)$, then for the attacker, $r_{li}, l \in 1, 2$ will be computed from $r_{lj}, (j \neq i, k)$ and $l \in 1, 2$ plus or minus a random value $(s_{lk}, l \in 1, 2)$ that is uniformly distributed over $\mathbb{Z}_q$. Based on the Assumption 1, it is not possible for an attacker with partial knowledge of $s_{lj}, l \in 1, 2$ to distinguish $r_{li}, l \in 1, 2$ from random. □

In large-scale auditing scenarios (e.g. with more than 100 individual auditors), colluding a few number (e.g. 10) of auditors will not have significant impact on the overall computation of fairness. Shansabadi et al. [44] empirically proved the detection of unfair behaviour of ML models is feasible in decentralised architectures, even in presence of malicious actors as auditors. Thus, the attacker should infer all the reconstructed public keys using the private keys it already collected from colluded individual auditors. If the attacker succeeds in doing so, then it will be in the position to generate parts of the audit cryptogram ($crypto_{1i} = g^{r_{1i}.s_{1i}} \times (\alpha_{1i})^{C_i} \times (\beta_{1i})^{\gamma_i}$ and $crypto_{2i} = g^{r_{2i}.s_{2i}} \times (\alpha_{2i})^{C_i} \times (\beta_{2i})^{\gamma_i}$ where $C_i = g^{p_i}$ and $p_i \in 2^0, 2^m, \cdots, 2^{7.m}$ where $m \in \mathbb{N}$ *and* $2^m > n$). Considering the DDH assumptions (Assumption 1), it will be impossible for attacker to calculate the reconstructed public key (r_{li}, where $l \in 1, 2$) without having access to *all* private keys. Thus, for the successful collusion of the auditing process, the attacker should bribe all the individual auditors. The public verifiability of FaaS protocol in the computations (which is available through fairness board) contains guarantees through ZKPs to ensure the FCS cannot misbehave in computation of the fairness. Moreover, any modification on the data in fairness board will be detected as it contains the ZKPs for ownership of private keys and the multiplication of the received audit cryptograms (as proved in Proposition 2). As a result of the Proposition 4, it will not be possible for the attacker to conduct a large-scale collusion attack by bribing a few individual auditors.

In summary, FaaS provides security, privacy and anonymity for the individual auditors by providing random-looking cryptograms recorded on the FB. An attacker will not be able to distinguish meaningful information from the FB and modify the recorded data on it without being detected. The universal verifiability through end-to-end adoption of ZKPs and cryptographic schemes provides dispute-free environment in which every one has the opportunity to re-compute the fairness metrics and ensure accountability of the FaaS system. Finally, the attacker will not be able to implement a large-scale collusion attack (and consequently, generate fake audit cryptograms to poison the process) since it needs to have access to all the private/public key combinations of the individual auditors.

7 Conclusion and future work

In this paper, we proposed FaaS as an end-to-end verifiable privacy-preserving protocol to audit fairness for ML systems. We implemented our protocol using off-the-shelf tools and achieved promising results. There are areas that can consider as future work. Our FasS protocol only verifies the computations of the fairness metrics. It does not support the initial training of the ML system for a fair outcome and hence assumes that the ML system is honest. In order to be able to offer FaaS without such an assumption, we plan to design

and implement a pre-deployment round of fair commitments from the ML model. In this paper, we only focus on the technical aspects of our protocol by proposing, implementing and evaluating it. In the future, we would like to conduct user studies around this topic and the user perception of the proposed protocol. Finally, the implementation of a real-world trustworthy fairness board is challenging. People might not trust a website owned by a particular entity, and protecting such a website against potential attacks is essential. To tackle this, blockchain-based architectures can be utilised [33, 34] to implement a secure tamper-proof fairness board. This is beyond the scope of this paper, and we leave it as future work.

Acknowledgements The authors in this project have been funded by UK EPSRC grant "FinTrust: Trust Engineering for the Financial Industry" under grant number EP/R033595/1, and UK EPSRC grant "AGENCY: Assuring Citizen Agency in a World with Complex Online Harms" under grant EP/W032481/1 and PETRAS National Centre of Excellence for IoT Systems Cybersecurity, which has been funded by the UK EPSRC under grant number EP/S035362/1.

Data availability Materials described in the manuscript, including all relevant raw data, will be freely available for any researcher wishing to use them for non-commercial purposes, without breaching participant confidentiality.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval This research has been conducted with no human involvement for data collection. The project obtained the ethical approval from the authors' institution.

References

1. Adler, P., Falk, C., Friedler, S.A., Nix, T., Rybeck, G., Scheidegger, C., Smith, B., Venkatasubramanian, S.: Auditing black-box models for indirect influence. Knowl. Inf. Syst. **54**(1), 95–122 (2018)
2. Aitken, M., Ng, M., Toreini, E., van Moorsel, A., Coopamootoo, K.P., Elliott, K.: Keeping it human: a focus group study of public attitudes towards ai in banking. In: Computer Security: ESORICS 2020 International Workshops, DETIPS, DeSECSys, MPS, and SPOSE, Guildford, UK, September 17–18, 2020, Revised Selected Papers 25, pp. 21–38. Springer (2020)
3. Aitken, M., Toreini, E., Carmichael, P., Coopamootoo, K., Elliott, K., van Moorsel, A.: Establishing a social licence for financial technology: reflections on the role of the private sector in pursuing ethical data practices. Big Data Soc. **7**(1), 2053951720908892 (2020)
4. Anderson, R.: Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley (2020)
5. Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., Rudin, C.: Learning certifiably optimal rule lists for categorical data. arXiv preprint arXiv:1704.01701 (2017)
6. Azad, M.A., Bag, S., Parkinson, S., Hao, F.: Trustvote: privacy-preserving node ranking in vehicular networks. IEEE Internet Things J. **6**(4), 5878–5891 (2018)
7. Azimi, V., Zaydman, M.A.: Optimizing equity: working towards fair machine learning algorithms in laboratory medicine. J. Appl. Lab. Med. **8**(1), 113–128 (2023)
8. Bacciarelli, A.: The toronto declaration: Protecting the right to equality and non-discrimination in machine learning systems (2023)
9. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283 (2001)
10. Brasher, E.A.: Addressing the failure of anonymization: guidance from the european union's general data protection regulation. Colum. Bus. L. Rev. p. 209 (2018)
11. Chouldechova, A.: Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. Big data **5**(2), 153–163 (2017)
12. Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., Huq, A.: Algorithmic decision making and the cost of fairness. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 797–806. ACM (2017)
13. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Annual International Cryptology Conference, pp. 174–187. Springer (1994)
14. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. Eur. Trans. Telecommun. **8**(5), 481–490 (1997)
15. De Cristofaro, E.: An overview of privacy in machine learning. arXiv preprint arXiv:2005.08679 (2020)
16. De Cristofaro, E.: A critical overview of privacy in machine learning. IEEE Secur. Privacy **19**(4), 19–27 (2021)
17. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 259–268. ACM (2015)
18. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the Theory and Application of Cryptographic Techniques, pp. 186–194. Springer (1986)
19. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 1–42 (2018)
20. Hao, F., Kreeger, M.N., Randell, B., Clarke, D., Shahandashti, S.F., Lee, P.H.J.: Every vote counts: Ensuring integrity in large-scale electronic voting. In: 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14) (2014)
21. Hao, F., Ryan, P.Y., Zieliński, P.: Anonymous voting by two-round public discussion. IET Inf. Secur. **4**(2), 62–67 (2010)

22. Hardt, M., Price, E., Srebro, N., et al.: Equality of opportunity in supervised learning. In: Advances in Neural Information Processing Systems, pp. 3315–3323 (2016)
23. Hu, H., Liu, Y., Wang, Z., Lan, C.: A distributed fair machine learning framework with private demographic data protection. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 1102–1107. IEEE (2019)
24. Jagielski, M., Kearns, M., Mao, J., Oprea, A., Roth, A., Sharifi-Malvajerdi, S., Ullman, J.: Differentially private fair learning. In: International Conference on Machine Learning, pp. 3000–3008 (2019)
25. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press (2014)
26. Kilbertus, N., Gascon, A., Kusner, M., Veale, M., Gummadi, K.P., Weller, A.: Blind justice: Fairness with encrypted sensitive attributes. In: 35th International Conference on Machine Learning, pp. 2630–2639 (2018)
27. Kusner, M.J., Loftus, J., Russell, C., Silva, R.: Counterfactual fairness. Adv. Neural Inf. Process. Syst. **30** (2017)
28. Larson, J., Mattu, S., Kirchner, L., Angwin, J.: How we analyzed the COMPAS recidivism algorithm. ProPublica 5 **9**(1) (2016)
29. Liu, J., Yu, F., Song, L.: A systematic investigation on the research publications that have used the medical expenditure panel survey (MEPS) data through a bibliometrics approach. Library Hi Tech (2020)
30. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. **30** (2017)
31. Mahdawi, A.: It's not just a-levels—algorithms have a nightmarish new power over our lives. The Guardian (2020)
32. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. Acad. Manag. Rev. **20**(3), 709–734 (1995)
33. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. In: International Conference on Financial Cryptography and Data Security, pp. 357–375. Springer (2017)
34. McCorry, P., Toreini, E., Mehrnezhad, M.: Removing trusted tallying authorities. School of Computing Science Technical Report Series (2016)
35. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. ACM Comput. Surv. **54**(6), 1–35 (2021)
36. Narayanan, A.: Translation tutorial: 21 fairness definitions and their politics. In: Proceedings Conference on Fairness Accountability, and Transparency, New York, USA (2018)
37. Panigutti, C., Perotti, A., Panisson, A., Bajardi, P., Pedreschi, D.: Fairlens: auditing black-box clinical decision support systems. Inf. Process. Manag. **58**(5), 102657 (2021)
38. Panigutti, C., Perotti, A., Pedreschi, D.: Doctor xai: an ontology-based approach to black-box sequential data classification explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 629–639 (2020)
39. Park, S., Kim, S., Lim, Y.s.: Fairness audit of machine learning models with confidential computing. In: Proceedings of the ACM Web Conference 2022, pp. 3488–3499 (2022)
40. Reuters: Amazon ditched AI recruiting tool that favored men for technical jobs. The Guardian (2018)
41. Richards, L.E., Raff, E., Matuszek, C.: Measuring equality in machine learning security defenses. arXiv preprint arXiv:2302.08973 (2023)
42. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
43. Segal, S., Adi, Y., Pinkas, B., Baum, C., Ganesh, C., Keshet, J.: Fairness in the eyes of the data: Certifying machine-learning models. In: Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society, pp. 926–935 (2021)
44. Shahin Shamsabadi, A., Yaghini, M., Dullerud, N., Wyllie, S., Aïvodji, U., Alaagib, A., Gambs, S., Papernot, N.: Washing the unwashable : On the (im)possibility of fairwashing detection. In: Koyejo, S., Mohamed,, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 14170–14182. Curran Associates, Inc (2022)
45. Shamsabadi, A.S., Wyllie, S.C., Franzese, N., Dullerud, N., Gambs, S., Papernot, N., Wang, X., Weller, A.: Confidential-PROFITT: confidential PROof of fair training of trees. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=iIfDQVyuFD
46. Siau, K., Wang, W.: Building trust in artificial intelligence, machine learning, and robotics. Cutter Bus. Technol. J. **31**(2), 47–53 (2018)
47. Stinson, D.R., Paterson, M.: Cryptography: Theory and Practice. CRC Press (2018)
48. Toreini, E., Aitken, M., Coopamootoo, K., Elliott, K., Zelaya, C.G., van Moorsel, A.: The relationship between trust in AI and trustworthy machine learning technologies. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 272–283 (2020)
49. Veale, M., Binns, R.: Fairer machine learning in the real world: mitigating discrimination without collecting sensitive data. Big Data Soc. **4**(2), 2053951717743530 (2017)
50. Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., MacNeille, P.: A Bayesian framework for learning rule sets for interpretable classification. J. Mach. Learn. Res. **18**(1), 2357–2393 (2017)
51. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 1–19 (2019)