IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 September 2022, accepted 14 October 2022, date of publication 19 October 2022, date of current version 28 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3215660

RESEARCH ARTICLE

Blockchain and NFTs for Trusted Ownership, Trading, and Access of AI Models

AMMAR BATTAH[1], MOHAMMAD MADINE[1], IBRAR YAQOOB[1], (Senior Member, IEEE), KHALED SALAH[1], (Senior Member, IEEE), HAYA R. HASAN[1], AND RAJA JAYARAMAN[2]

[1]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
[2]Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, United Arab Emirates

Corresponding author: Ibrar Yaqoob (ibrar.yaqoob@ku.ac.ae)

This publication is based upon work supported by the Khalifa University of Science and Technology under Awards No. RCII-2019-002–Center for Digital Supply Chain and Operations Management, and CIRA-2019-001.

ABSTRACT The demand for high-quality Artificial Intelligence (AI) models is ever-increasing in this digital era. However, most of the existing methods leveraged for managing the ownership, trading, and access of AI models fall short of providing traceability, transparency, audit, security, and trustful features. In this paper, we propose a solution based on blockchain and Non-fungible Tokens (NFTs) to manage ownership rights and exchange of AI models in a transparent, traceable, auditable, secure, and trustworthy manner. Smart contracts are employed to enforce ownership, ease of access, and exchange policies for the unique NFT linked to an AI model. We use decentralized storage of the InterPlanetary File System (IPFS) and proxy re-encryption oracles to securely fetch, store, and share data related to AI models. We present algorithms along with their implementation, testing, and validation details. The proposed solution is evaluated using cost and security analyses to show its affordability and resiliency against security threats and attacks. All smart contract codes are made publicly available on GitHub.

INDEX TERMS Blockchain, decentralized storage, non-fungible tokens (NFTs), oracles, provenance, proxy re-encryption, smart contracts.

I. INTRODUCTION

Artificial Intelligence (AI) is constantly evolving in its mission to mimic human intelligence. The adoption rates of AI in different domains, such as healthcare, finance, and transport, are consistently increasing. There have been many challenges posed since the inception of AI [1], [2]. For example, AI models and datasets are scattered among individuals, organizations, and companies, hindering the potential of AI and creating a barrier to further contributions [3]. Therefore, it is believed that shifting toward a transparent collaboration model can boost AI development and potential.

The challenges encountered in using AI systems are attributed to the current development process and the complexity of the domains that use the system. For example, there are different data types in the healthcare domain, such as Electronic Health Records (EHR). Each entity can have different EHR systems with irregular standards to encode events. In addition, the data itself is heterogeneous due to the uniqueness of patients and their cases, and this applies to behavioral data, medical images, physiologic data, and environmental data [4]. Similar complexities exist in fields such as finance and transportation [5], [6].

The centralized nature in which AI development is typically carried out results in the creation of AI assets that are highly specific and narrow in applicability. The root cause of the centralized status quo of AI research, which behaves as a barrier to individuals, is threefold: 1) the lack of interoperability standards; 2) the lack of appropriate infrastructure for AI collaboration; and 3) the high cost of development and operation. It is common to see large entities such as Google, Amazon, Microsoft, and IBM as the forward-facing AI contributors [7], [8]. At the same time, smaller institutions, researchers, and individuals find no major incentives to build solutions for end-users. There is a need for a system that allows the participation of different entities in the AI process.

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

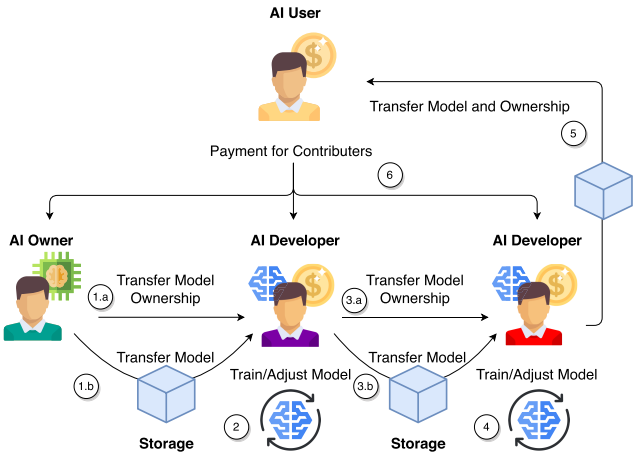

FIGURE 1. The evolution of an AI model whereby developers contribute to the model shared by an AI owner to provide an AI user with the required model.

The difficulty of using AI as a third party is that it has a black box construct that provides no explainability for the user. In addition, the lack of information on the source of data, training process, and validation techniques of the model makes it harder for the user to trust the AI system. Therefore, the system must track information regarding all assets and provide provenance of data to overcome trust concerns. Tracking data provenance and ensuring transparency, in turn, 1) increases trustworthiness 2) enables more efficient use of AI models due to a better understanding 3) improves cooperation and collaboration by assisting entities in making confident decisions when exchanging AI assets. Tracking AI assets is crucial but is a point of concern with traditional centralized approaches with a single point of trust and failure. As such, utilizing Distributed Ledger Technology (DLT) can provide the needed groundwork for building the desired decentralized system. Blockchain technology is a special type of DLT that provides immutability, transparency, auditability, and decentralization, all of which are required for operating such a marketplace. By using blockchain technology, the history of how the model was made and improved is kept in a distributed ledger that acts as a chain of provenance for the AI assets.

It is important to protect the ownership rights of the AI assets and their chain of original creators and contributors to prevent users from falsely claiming rights over others' works. Non-fungible Tokens (NFTs) are a powerful mechanism that sits on top of the blockchain network and provides a means to represent assets as unique tokens that are associated with their appropriate owners [9]. NFTs were introduced with Ethereum's improvement proposals and became a standard under ERC-721 (Ethereum Request for Comments 721). Unlike the fungible tokens standard (ERC-20) commonly used for cryptocurrencies, NFTs are unique and non-interchangeable. At the same time, NFTs reserve the right to use and distribute assets and manage them in public records with indisputable proof of ownership to ensure appropriate remuneration. This is because the unique token can be transferred while keeping an immutable history of a digital asset's information [10], [11]. However, in a lot of cases, users prefer to keep their assets private, which is what we consider in our solution. Keeping the assets completely private without validation can leave the door open to inaccurate information and deception. Therefore, we utilize Trusted Execution Environments (TEE) and blockchain oracles to aid the end-users in validating assets while preserving the privacy of the assets [12]. Furthermore, the storage of the NFT is usually done on a decentralized storage network. The asset is uploaded along with the corresponding metadata, and both are persistently saved on the blockchain. Finally, the system has to ensure fair compensation for AI developers in return for providing the appropriate AI assets to users. AI assets can be models, datasets, or scripts used to adjust them.

Our proposed approach addresses AI collaboration by emphasizing AI models and how developers can enhance them. Contrary to the existing AI collaboration research that mainly focuses on creating a GDPR-compliant (General Data Protection Regulation) data pool from multiple confidential datasets [13]. AI stakeholders can freely contribute by sharing models and datasets, improving models, or securing the network. Our proposed system focuses on transfer learning and fine-tuning problems where the models can be enhanced and repurposed to achieve the desired goals. Moreover, this is without the assumption of possessing and publicly sharing private datasets by the owner. The AI system workflow can be seen in Figure 1, where the marketplace provides payment flexibility through royalties for all the participating developers that have contributed to the creation of the final model. As a result, the marketplace contributes to the AI field by making assets readily available to the average user, thereby increasing the technology's footprint. AI models are IPs that must be protected and compensated appropriately for their use. Furthermore, NFTs help keep track of provenance data for AI assets, facilitating collaboration and exchange while preserving the ownership of assets. Finally, smart contracts aided by TEE and oracles realize asset validity and ensure proper rule enforcement. The main contributions are as follows:

- We showcase how blockchain and NFTs can be leveraged to manage the ownership, trading, and access of AI models in a manner that is decentralized, transparent, traceable, secure, and trustworthy.
- We develop and implement smart contracts incorporating NFTs, decentralized storage, and proxy re-encryption (PRE) oracles to govern the interactions between entities in the network with no central entity. All code developed is made public on GitHub.[1]
- We define a method for the provenance of AI models on the blockchain to achieve transparency and explainability for the user while preserving the rights of owners

[1] https://github.com/AINFTProject/AI_NFT

TABLE 1. Summary of the existing solutions.

Existing Works	AI Technique	Network Model	Goal
Intelligence via Blockchain [14]	Federated Learning	NA	AI Model Provenance
Tracking of Artificial Intelligence Assets [15]	Generic	Public	AI Assets/Value chain Provenance
Blockchain Enabled AI Marketplace [16]	Transfer Learning	Private	Address Transfer Problem issues
SAIaaS [17]	TEE Execution	Public	Secure Cloud Execution for AI
Machine Learning Meets Blockchain [18]	Federated Learning	NA	Privacy in Distributed AI
A Marketplace for Trading AI Models IoT Data [19]	Federated Learning	Consortium	Trusted Collaborative AI
FDPDDL [20]	Collaborative Differential Privacy	Decentralized P2P	Fair and Private AI
Towards Industrial Private AI [21]	FL + Homomorphic Encryption	NA	AI Data and Model Security

and developers through a royalty scheme for appropriate compensation.
- We present eight algorithms along with their full implementation, testing, and validation details. We validate the smart contracts through a detailed use case with exception handling and assertions. We present the cost and security analyses to show the effectiveness of our solution.

The organization of the paper is as follows. Section II presents the related work. Section III presents the background. Section IV provides an overview of the proposed approach and highlights the architecture and sequence of interactions. Section V discusses the implementation details and verifies the validity of the solution. Section VI presents the cost and security analyses of the proposed solution. We provide concluding remarks in Section VII.

II. RELATED WORKS

The development of AI continues to be mainly a centralized process, which impedes its advancements. In this regard, recent research focuses on facilitating the sharing of AI models among developers to increase the workforce and advance the progression pace. The traditional and most common convention for AI model sharing is cloud-based, which is criticized for issues such as centralization that impede trust and privacy. One of the recent works proposes a solution to tackle the centralization issue found in cloud-based marketplaces for AI by using virtual premises to enforce privacy policies and access rules [22]. However, such a solution does not provide complete trust as a decentralized network.

The ongoing research is directed toward using blockchain to decentralize traditional central approaches, as it is considered the most optimal solution for immutability and transparency. For example, ProvChain builds a solution for the provenance of cloud data and assets using blockchain to create tamper-proof records. However, a drawback of this solution is the need to establish trust with the cloud provider as a requirement to store data [23]. Another method proposes a solution for distributed machine learning while maintaining privacy by establishing a network of local data holders and compute nodes that collaborate to compute gradients that are aggregated to generate a global gradient [18]. The work focuses on privacy and security with the inclusion of differential privacy and constantly appending to the blockchain with each iteration, raising concerns regarding performance and cost. Finally, a different proposal solves the issue of the difference in data for validation coming from distributed sources by utilizing a private blockchain to govern organizational data providers in [16].

For AI models trained in discrete phases, the research found FL to be a great tool to assist with the distributed development of machine learning models. One of the proposed approaches leverages blockchain to act as a provenance foundation that contains metadata about the AI models, such as training data, training process, tests, and results. While a fusion manager aggregates the computed weights of each node and appends them to the blockchain [14]. Another solution extends the previous work by tracking the entire AI value chain rather than focusing on generating global models through FL. The value chains involve organizational interactions that involve dataset exchange [15]. The solution uses a public blockchain and private-public keypairs for selective access control. Although both approaches describe the provenance of the assets, they do not provide or implement a scheme for exchanging and purchasing models.

A two-stage scheme is proposed in [20] to produce more accurate models while maintaining privacy and ensuring fairness. The first stage relies on a Differentially Private Generative Adversarial Network (DPGAN) to set credibility and tokens for peers, and in the second stage, a Differentially Private SGD (DPSGD) is used for collaborative deep learning training. During that time, credibility and tokens are updated to reflect participants' contributions, to ensure fair compensation. The proposed approach focuses on fairness, with reward being equivalent to the contribution and privacy of the model and data since parties are more inclined to contribute in a private setting. The study conducted in [21] focuses on securing both the data and the model through homomorphic encryption. The approach is designed such that in untrusted environments, the parameters are encrypted but can be used in computations. The output of the model is encrypted, and only trusted parties with the secret key can decrypt it. As for the images used, a chaotic system is used for scrambling the image and a discrete wavelet transform (DWT) for encoding. Finally, some noise is added for increased security. To reverse this process, a denoising convolutional neural network (DCNN) is utilized. While both works attempted to preserve the privacy of the model parameters and data, the study conducted in [20] is limited in applicability as only gradients can be exchanged with no global model. On the

other hand, the approach proposed in [21] is in its infancy and it is difficult to apply homomorphism for complicated operations.

One category of solutions for AI collaboration aims to build a fair marketplace for data providers, developers of models, and computation devices. For instance, Secure AI as-a-Service (SAIaaS) is a solution that allows users to send their data to run their AI training in a secured cloud environment and interact with providers of other datasets, models, and compute power [17]. In addition, the solution encourages collaboration among the users in the marketplace using a rewarding system. Another solution integrates FL with Internet-of-Things (IoT) devices to establish a network of devices that collaborate on training models [19]. The approach leverages blockchain to govern the network and rewards the owners of the devices with fair payments. The authors detail how mining nodes back IoT devices to handle computation, and on each request to train a model, the model performance gets updated based on the most accurate result. A concern of the solution is the assumption that a miner is always available, which might not be the case. A summary of the technique, network model, and goal of the addressed literature can be seen in Table 1.

III. BACKGROUND

In this section, we present an overview of technologies and concepts utilized in designing our proposed solution. We illustrate how traceability and provenance are achieved. Furthermore, we clarify the roles of TEE, NFTs, and decentralized storage in our solution.

A. BLOCKCHAIN AND SMART CONTRACTS

Blockchain technology constructs a decentralized ledger by utilizing hash functions to store blocks of transactions immutably. A hash function performs a one-way cryptographic mapping of data of an indeterminate size to a fixed-size hash value, where swapping one bit in the input completely changes the hash value [24]. Transactions in a blockchain are grouped into blocks, and each transaction represents an interaction between two entities. The blocks are hashed, and the resulting value from each block is incorporated inside the following block's metadata to form a chain of immutable data [25]. The transactions and blocks are validated and published by mining nodes that run a consensus algorithm. The most common consensus algorithm is Proof-of-Work (PoW), which requires solving a computationally difficult puzzle [26], and the solution gets embedded into each block. Entities on the network can keep their transactions as long as they keep the private keys to their accounts. These keys are used to sign the transactions and make sure they are being sent by the party who claims to be sending them.

In our proposed solution, a public permissionless blockchain is utilized as a provenance chain that maintains the history of a model. This type of blockchain helps remove the prevalent entry barrier to AI technology. Furthermore, the ease of participation helps secure the network by including more competing and collaborating participants in the consensus process. Blockchain also serves as a decentralized financial medium between model owners and users. Blockchain integration depends on smart contracts, which govern the interactions. The concept of a smart contract was first introduced in the 1990s as a computerized transaction protocol that executes the terms of a contract [27]. A smart contract is used to secure relationships on a network to eliminate the need for trusted intermediaries and prevent malicious transactions [28]. It utilizes a deterministic language that can be used to define transactions, process inputs, write outputs, change state variables, and manage access control [29]. In the proposed solution, ownership, auctioning, royalty distribution, and the governance of model assessment are done on-chain. In contrast, the storage of the model, metadata, and assessment results are done off-chain.

B. TRACEABILITY AND ASSESSMENT

Transparency, auditability, and provenance are essential components for building trust. We leverage the blockchain to achieve the features above while also providing the ability to track system transactions. As a result, the common ambiguity in the AI field is eliminated, and the spread of AI development is facilitated. Directional Acyclic Graph (DAG) is a mechanism that we use to maintain the history of AI assets and, by extension, their provenance. DAG uses a parent-child association between assets. Moreover, DAG keeps track of what operations were performed on the parent to get the child. Some solutions incorporated DAG to represent AI assets, including models, datasets, and operations [14], [15]. Our solution leverages the presented notion in the literature to construct the provenance graph on the same types of assets. However, we condense the representation of the three entities into a single compound entity on the blockchain to align with our approach. Moreover, further details of the compound entity are stored off the chain.

A critical part of maintaining provenance and establishing trust is validating the AI model, ensuring the performance results align with what is claimed. The adverse goals of validating and assessing a model without violating the confidentiality of the model and data prove to be challenging. This has resulted in proposing several methods for satisfying the requirements of AI collaboration without breaching privacy. But as mentioned, most approaches are applied to data, such as homomorphic encryption and differential privacy, or approaches that use collaboration methods that preserve data privacy but expose the model, such as FL and Secure Multi-party Computation (SMPC). On the other hand, TEE schemes for private inference have proven to perform well, especially in comparison to cryptographic techniques [30]. TEEs give a guarantee of confidentiality in an untrusted environment through hardware and software protection measures, as well as the integrity of code execution through attestation [31]. The trusted compartments are called secure enclaves, which remote hosts can verify through

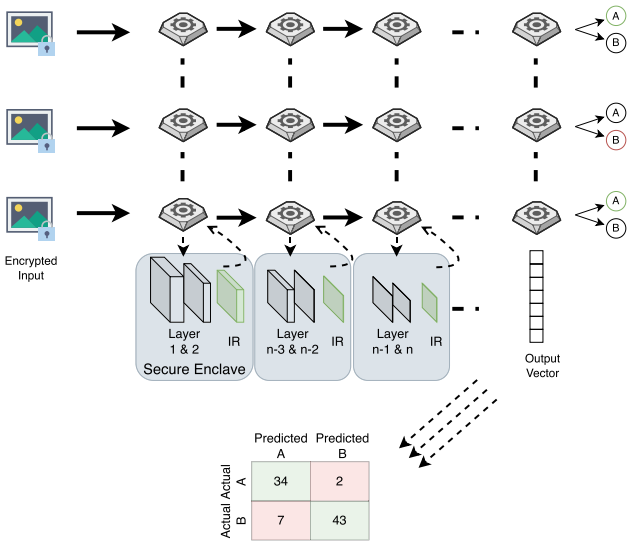

FIGURE 2. A network of worker oracles collaborates to evaluate a private model. In this case, a classification problem with two classes, A and B, is proposed, and the tester provides a small set to test the private model through the worker oracles. The result is aggregated from the workers on the smart contract.

remote attestation. Several providers offer different TEE solutions, such as Intel SGX (Security Guard Extensions), ARM Trustzone, and AMD PSP (Platform Security Processor) [31], [32], [33].

The cooperation of the oracles increases the number of participants due to low resource requirements. Moreover, partitioning the model solidifies security and trust due to the futility of breaching the TEE to attain a portion of the model, as it relies on other portions. The nature of the collaborating oracle workers can be seen in Figure 2. Furthermore, each oracle submits the encrypted intermediary result (IR) to the next worker. Before starting, remote attestation of the enclave must be conducted to establish the correct code in the enclave. Different providers may have slight differences in the attestation protocol, but Intel SGX remains the primary focus for the example scenario. The host with the TEE is the attestor, and the entity attempting to verify it is the challenger. Then the attestor's application requests initiation of the attestation process, starting with a local attestation from the application enclave. Afterward, the local attestation has to be verified by a quoting enclave, which signs the local attestation with its symmetric key after verification. The quote then gets transferred to the challenger through the attestor application to be verified. The signature of the quote can then be validated using the Attestation Verification Service offered by Intel [34].

C. NFTs AND DECENTRALIZED STORAGE

Since the inception of blockchain technology in 2008, the technology has matured significantly, consistently adding blocks on top of it by the community to enhance the technology [35]. Most of the advancements have come with the introduction of smart contracts, which in turn have enabled the utilization of tokens. Non-Fungible Tokens have recently gained huge traction in their application to solve a challenge faced across many different digital domains, which is proving ownership. Proving ownership in different applications and enforcing it throughout the system has been an issue. NFTs leveraging the blockchain surged to overcome such a challenge. By giving a unique tamper-resistant token to each AI asset, we can guarantee fair payment, enforce granular access control, and preserve the buyer and seller's rights. NFTs achieve this through transferability, immutability, transparency, availability, and fraud prevention [10], [11]. By keeping track of ownership, it is also possible to utilize a royalty scheme that protects the payment rights of contributors and creators. Royalties prove to be crucial in the proposed approach due to the transfer learning mode that it supports, which involves several contributors. The smart contract produces the NFT using the owner's address, the smart contract, and a contract-specific token to generate the unique NFT. It is common first to digitize the asset appropriately, store it in the appropriate medium, send a signed transaction to the smart contract, and mint the NFT to correspond to that owner in a record on the blockchain and distributed ledger after confirming the associated block [10].

The storage of the NFT is usually done on a decentralized storage network. The asset is uploaded along with the corresponding metadata, and both are persistently saved on the blockchain. This approach is necessary to avoid bloating the blockchain since the assets vary in size, and the data has to be saved across the network. Storing data on a decentralized storage system and maintaining an immutable link to it efficiently achieves the desired result. The storage can be public or private, which the owner controls by deciding to encrypt the asset before upload. Private assets are suitable for trading on the decentralized networks, while public ones apply to patented assets with an appropriate licensing scheme [11], [36].

IV. PROPOSED SOLUTION

We propose a blockchain and NFTs based solution to facilitate trusted AI operations between creators, developers, and users. The solution aims to enable ease of access, traceability, and fair payment while ensuring the privacy of the AI models. Furthermore, each participant in the solution is assigned a different task with an appropriate incentive to execute it honestly. The proposed solution can be seen as a marketplace where participants trade private AI assets.

A. SYSTEM DESIGN AND COMPONENTS

In the proposed solution, there are two main entities, which are the buyer and the seller. However, each assumes a different role based on the current function of the system. For example, an AI developer becomes an AI owner after purchasing and possessing a model while still being an AI developer. Furthermore, along with the end-users, several

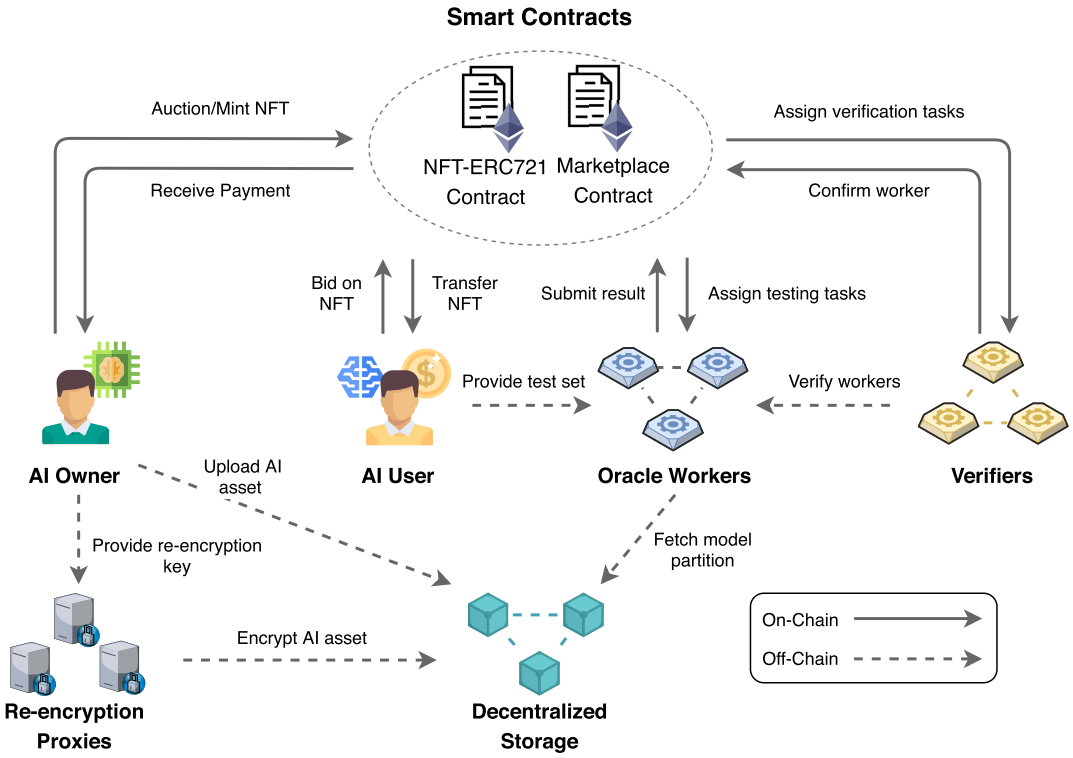

FIGURE 3. A system entity overview diagram highlighting the interactions for minting an NFT, asset upload, auctioning, verification, and assessment.

entities collaborate in the proposed architecture to achieve the requirements of the approach, as can be seen in Figure 3.

- **AI Creator/Owner**: The crux of the system is the creators and owners of AI assets. In a marketplace, the creator is the first entity that submits their model, proving to be the one that created the model or generated/collected the data. Owners are the subsequent AI model purchasers, with a corresponding NFT to prove ownership. The distinction between creator and owner is required for the right payment scheme and to enforce the appropriate policies. According to the agreed terms, owners and creators share their assets for the monetary incentive and model development. As for communication, the creator/owner would use the Dapp to communicate with the blockchain and execute requests.
- **AI User/Developer**: The user and the developer can also be generalized as entities in the market looking to purchase AI models. The user is an entity that is interested in using AI assets for different purposes. In comparison, the developer is an entity interested in gaining monetary compensation for their services and advancing AI technology. Furthermore, both entities can be AI owners when acquiring an AI asset. While the developer might be so temporarily, an AI user usually takes the model off the chain to be used personally. However, in both cases, the model trace remains until the last exchange on the blockchain. Such end-users also rely on DApps for on-chain communication.
- **Smart Contracts**: We develop smart contracts to handle auctions and payments; define access rules; enable the assignment of tasks for the different oracles; and perform a model assessment in an automated manner.
- **Decentralized Storage**: We use decentralized storage to complement the existing scheme. Specifically, the InterPlanetary File System (IPFS) is used to store data offchain [37], [38]. The features of IPFS complement the requirements of integrity, transparency, and availability. Furthermore, system users can enforce confidentiality on the storage system through encryption.
- **Oracles**: External sources that aid the blockchain in interacting with the outside world are the oracles of the system. In this work, oracle workers are responsible for assessing AI models. Verifiers handle the verification of the oracle workers to adhere to the set authentication criteria. It is essential to maintain the decentralization in the oracle subsystem, not to negate the existing features of the decentralized network, and to avoid the "Oracle Problem" of having a central data source that can lead to corrupt, malicious, or incorrect data [39]. Therefore, the oracle subsystems must reach a consensus on the correct result. Oracles get monetary compensation for executing their tasks successfully. In addition, entities requesting tasks compensate the oracles appropriately, such as model assessment tasks. As with the other entities, oracles have to register with a unique Ethereum

Address (EA) on the blockchain and then get authenticated before participating.

- **Proxy Re-encryption (PRE)**: We use PRE to re-encrypt the partitioned AI model to securely transfer it to the worker oracles. These nodes delegate the process while ensuring a secure transaction between the assessment oracles and the owner of the model. Such a network can be opted out of if the owner selects to do the re-encryption. We mainly consider a threshold proxy re-encryption scheme such as the one proposed by the NuCypher suite [40]. Generally, the owner generates a re-encryption key using his secret key and the receiver's public key (the oracle in this case). The PRE encrypts the partitioned model's symmetric key, which can then only be decrypted by the intended oracle's private key. We also refer to the proxy as a Re-encryption proxy (REP).

B. SYSTEM INTERACTIONS

Herein, we explore the proposed system interactions. Edge cases are limited for readability but revisited in the validation section. We distribute the interactions into three sections, and overlapping sections do not have repeated interactions but rather are condensed and could refer to existing sections. The first sequence pertains to the model's upload, auction, and transfer, which can be seen in Figure 4.

1) MODEL UPLOAD AND EXCHANGE

1) The process starts with uploading the asset along with the metadata, which includes all sorts of information possible to help provide explainability to the users. An owner/creator uploads the data on the IPFS, so any change in the data invalidates the hash. Furthermore, the path is an IPNS (InterPlanetary Name System) link [41], allowing changes to happen while keeping track of the asset. However, changes not tracked on the blockchain are deemed invalid since the IPFS hash on the blockchain is different than the one pointed to by the IPNS unless it is updated.
2) The Owner/Creator uploads the path to the uploaded data and, if valid, mints it as an NFT on the NFT smart contract. Doing so binds the asset with a token ID and the smart contract used, enabling exclusive access for ownership of the asset on the blockchain. Furthermore, when minting the asset, the owner/creator can identify any parent assets that contributed to the current asset. They are incentivized to transfer the credibility and exposure of an already existing asset previously verified. Moreover, it is to build a provenance chain that users can utilize to establish trust for the new asset. The owner can also set the royalty they wish to receive for using assets within the set range.
3) The owner can auction the asset and set a period for it but can only do so after validating the ownership of the asset through on-chain access policies. The marketplace Smart contract carries out all the tasks of the auctions.
4) Events are emitted to the users about the new auction and can bid now. Any auction model can be implemented. In the given scenario, an English auction is conducted Figure 4, with every user required to bid higher than the previous one. An event is emitted after each new bid to notify participants about the auction state.
5) During the bidding process, only the outbid users can withdraw their bids, and the highest bidder is not allowed to do so. The smart contracts also keep track of the funds received and transferred, and policies are in place to prevent any double withdrawal.
6) The auction can end once the period is over by having a timing oracle invoke the end of the period or by the asset's owner if he is satisfied with the current highest bid. Once the auction is over, a winner is announced and registered as the asset buyer. Then the buyer gets notified through an event to collect the asset. Finally, the participants get notified about the end of the auction so they can withdraw their corresponding bids.
7) The owner calls a function to ask for the NFT ownership to be transferred to the buyer to receive the payment. The Marketplace smart contract contacts the NFT SC to transfer the NFT. The NFT contract transfers the ownership and computes the royalties for the contributors of the asset.
8) After appropriately assigning the shares to each of the concerned entities, the shares can be withdrawn from the marketplace contract by their assigned address only.

2) MODEL ASSESSMENT

The sequence of the aforementioned interactions is under the assumption of a validated and assessed model. However, the model undergoes an off-chain process to assess and validate the provenance data that has not been recorded on-chain, as detailed in Figure 5.

1) A user needs to have a valid TEE to participate as a worker. To do so, the user must provide a remote attestation for his/her device. When the enclave is instantiated, a measurement of the enclave is taken as a record of its identity. Remote attestation is done by first going through local attestation, followed by the generation of a quote by the Quote enclave.
2) When a user attempts to register as a worker, they submit their quote and signature. The challenger has already sent a nonce, which is included in the quote to make sure that the attestation is up-to-date.
3) Verifiers act as the challengers that verify the quote, and they can register as long as there is no conflict of interest by assuming another role in the network. Although there are several verification methods, this scenario assumes the verifiers validate quotes with the Attestation Verification Service [42].

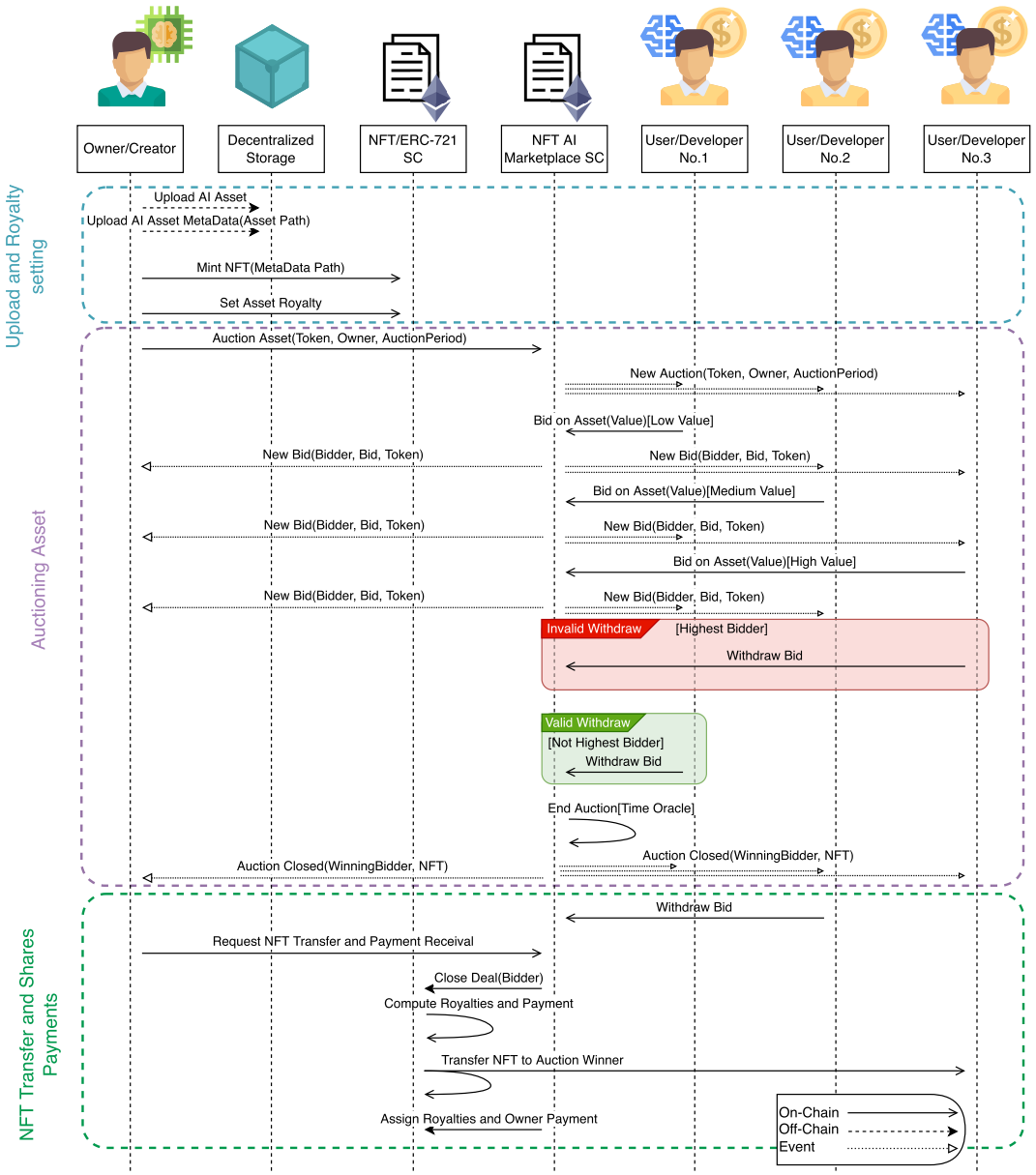

FIGURE 4. A sequence diagram highlighting 1) Upload of model and metadata 2) Auctioning of the AI asset 3) Transfer the NFT and distribution of shares for contributors based on royalties.

4) Several verifiers submit their verification results until they reach a consensus on the smart contract. Then, based on the consensus, the registering user is either rejected or registered.
5) The network workers can now register for model verification tasks based on available verification requests. The request can be initiated by a user interested in the model, a developer who wants to validate the results, or any entity testing how appropriate a dataset is.
6) To separate concerns and maintain secure operations, each worker can only handle one task at a time. Once the workers are assigned, their keys are shared with the owner/delegates to establish a secure connection. Before moving on, the owner conducts a remote authentication as the challenger. Within the attestation, the public key of the enclave is shared. Prior to exchanging any secrets, the owner must validate the signer of the enclave (MRSIGNER) and the enclave attributes to ensure the provisioning of data to the appropriate unaltered enclave [34].
7) The owner generates the fragmented re-encryption keys and distributes them to the re-encryption proxies. The proxies can then re-encrypt the key of the partitioned model that the owner uploaded. The enclave's public key is used to generate a re-encryption key, so even the worker cannot decrypt the secret.

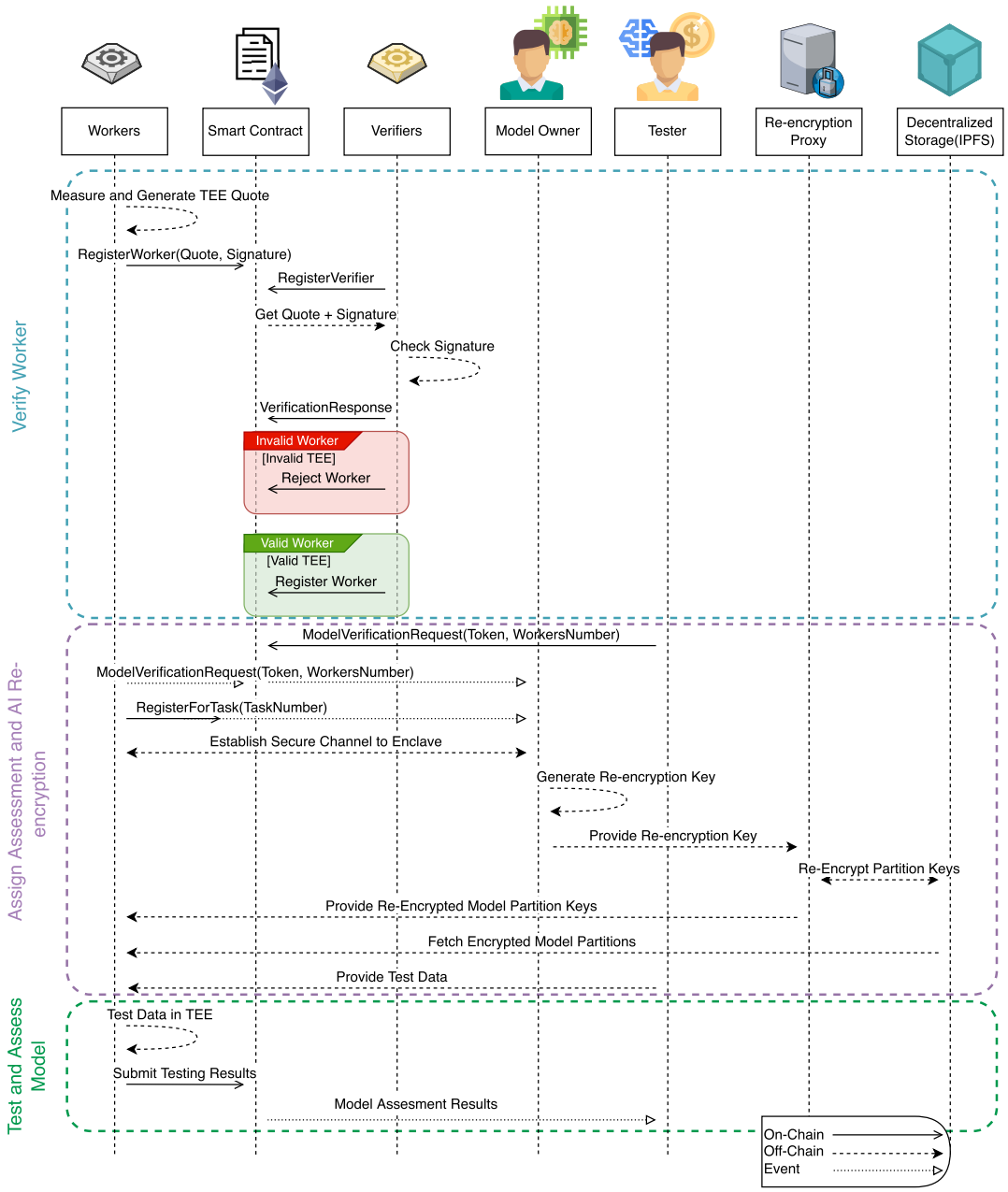

FIGURE 5. A sequence diagram highlighting: 1) Verification of workers with valid TEE 2) Assignment of tasks to workers for assessment and re-encrypting the model partitions 3) Testing and assessing the AI model based on the aggregated results.

8) Workers fetch the encrypted model partitions and test them inside the TEE. Each worker submits the result to the next worker. Several worker lines work in parallel based on the tester's conditions and requirements. As an added layer of security, the tester can use one of the approaches in the literature to privatize their small validation set. An example is data anonymization to prevent personal data disclosure and maintain privacy [43].
9) After the testing goes through the whole process and the final result is with the worker's head, the heads submit their results to the smart contract to reach consensus and provide the model assessment. Generally, the results depend on the testing goal of a model. However, test results generally consist of a confusion matrix from which several evaluation metrics can be derived, such as accuracy, precision, and recall.

3) TRADING AND PAYMENT

Figure 6 shows a scenario with two AI developers that wish to enhance an existing model before reselling it for a profit

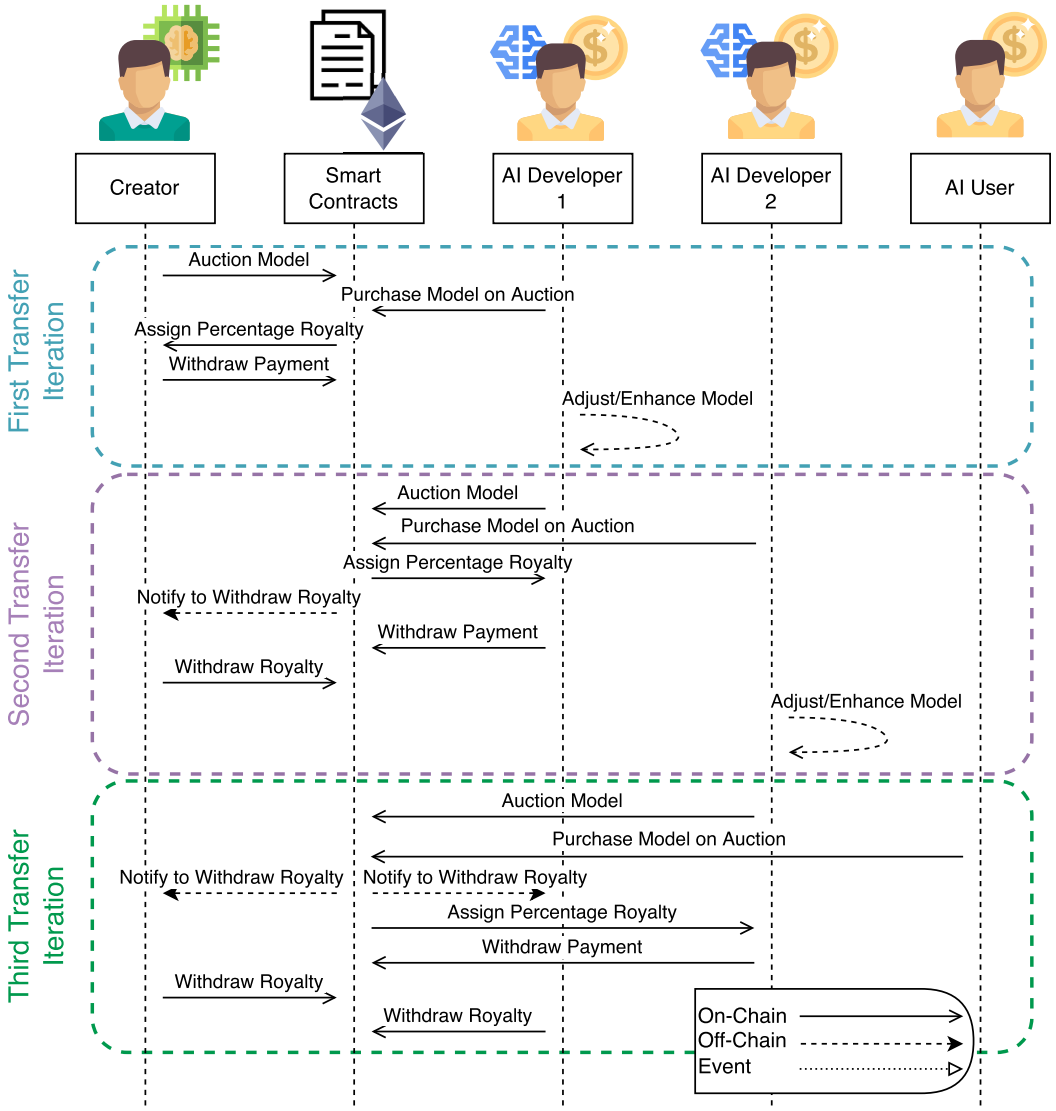

FIGURE 6. Model exchange scenario through the model evolution and transferring the NFT ownership to the new owner with every iteration through the smart contract.

and providing better AI services for the average user. On the other hand, the user is interested in utilizing the AI solution.

1) Starting with the original creator, the model is auctioned by the creator and bought by developer 1. The internal interactions are similar to what was described in "Auctioning Asset" and shown in Figure 4.
2) The creator is assigned a royalty percentage for future purchases of the model or its variants.
3) The creator withdraws the total price for the model since there are no other contributors.
4) Developer 1 then adjusts and enhances the model and then repeats the auctioning process. In this case, developer 2 purchases the model.
5) Developer 1 is now assigned a royalty percentage too. Once the transfer occurs, the smart contract assigns the majority of the paid amount to developer 1, while the creator gets a smaller share based on the royalty amount.
6) For the third iteration, the same process takes place, with developer 2 being one of the contributors now. This time, an AI user buys the model through the auction.
7) The smart contract notifies the creator and AI developer 1 to withdraw their royalties, while AI developer 2 receives the remaining payment for the model after the royalty distribution.
8) AI developer two is now assigned a royalty that gets triggered when the AI user sells the model, as well as the previous owners, the creator, and AI developer 1.

V. IMPLEMENTATION AND VALIDATION DETAILS

In this section, we present the implementation details of the proposed approach and outline the development of the proposed design. Two main smart contracts are constructed, with one managing the AI NFTs while the other acts as a marketplace that manages trades. We implement the NFT

Algorithm 1 Minting NFT and Setting Royalty of Asset

```
Input: ownerAddress, tokenID, uri, parent, parentToken
Require: msg.sender == ownerAddress
call: erc721.mint(ownerAddress, tokenID)
call: erc721.setTokenURI(tokenID, uri)
call: setToken(ownerAddress,
parent,tokenID,parentToken)
call: setRoyalty(ownerAddress, tokenID)
emit: ownerAddress,tokenId
```

contract to extend the ERC-721 smart contract and contain a compartment that governs oracle workers for AI model assessment. The marketplace contract holds the auctions for the NFTs and executes the trades. Since this is a specialized domain and not a general one, we set up a single contract that multiple users can use to manage the NFTs. This saves the users from having to do extra work. We make the smart contracts code publicly available on Github.

A. ALGORITHMS

The smart contracts were implemented on the Ethereum blockchain using the Solidity language. The system's main functions are detailed and interpreted in this section to clarify the approach's logic. We mostly represent the entities by structs (structures) and access them using the convention *struct*.*variable*. Since several entities have the same role simultaneously, mapping to the structs is created to access them using their key, such as *struct*[*key*].*variable*. Some variables that describe the properties of blocks and transactions are used in the algorithms, such as *msg*.*sender*, *msg*.*value* and *block*.*timestamp*. Their properties are provided globally by solidity, where *msg*.*sender* get the sender of the message, *block*.*timestamp* gets the current timestamp for the block, and *msg*.*value* is Ether (Wei) amount sent with the transaction. Variables preceded by a . indicate they are part of a structure.

Algorithm 1 is the fundamental function that adds the assets to the network, and it is the mint function that links the asset with an NFT on the blockchain. Next, the unique asset is submitted with a Uniform Resource Identifier (URI) and associated with the owner. Then, functions setToken( ) and setRoyalty( ) are called to link the token to an existing asset and set the appropriate royalty. Lastly, the function sends an event to inform participants that the owner now has a new NFT.

In Algorithm 2, we describe an internal function that brings the chain of contributors, mimicking a provenance chain, or reverts if there are no predecessors. The function creates an array, loops through the provenance chain from child to parent, and adds them to the array. Contributors receive their assigned shares based on the entries of the array.

Once the NFT is minted, royalty and provenance chain details are recorded accordingly, and the assets can be auctioned on the marketplace smart contract. To support that, a function on the NFT smart contract successfully

Algorithm 2 Get Existing Royalty Chain of NFT

```
Input: _token
if token[_token].childOf == address(0) then
    revert("No Royalties")
end
decayRange threshold of royalty chain length
royaltyAccounts is an array of addresses
parentAccount current address of parent
parentToken current token of parent
counter ← 0
parentAccount ← token[_token].childOf
parentToken ← token[_token].parentToken
while parentAccount != address(0) && currentLength <
  decayRange do
    royaltyAccounts[counter] ← token[_token].childOf
    parentAccount ← token[parentToken].childOf
    parentToken ← token[parentToken].parentToken
    counter ← counter +1
end
return: royaltyAccounts, counter
emit:royalyAccounts
```

transfers the NFT and distributes fair payments. For example, Algorithm 3 details the function for closing the deal once an auction ends in the marketplace smart contract. If there are no parents to the asset(*address*(0)), the whole payment goes to the owner. If there are, each entity has its share computed by the smart contract with dynamic royalties based on a specific decay factor. The royalty decreases when the entity keeps receiving them and more contributors join.

Even though the owner minted the asset, it must be verified to gain users' trust. The verification is possible using oracle workers, which the smart contract governs to verify the AI models. The workers can register for a verification task as shown in Algorithm 4. The function first ensures that 1) the required number of workers is unfilled and 2) the worker has not previously registered for the task. Then, the smart contract associates the worker with the task and adds the worker to the list. Finally, the task accounts for the number of workers registered. Line 7 is where the worker's address is added to the task array.

The workers verify the models and metadata and submit their results to the smart contract. The submission details can be seen in Algorithm 5. Since the workers are decentralized and test sequentially in parallel, only the oracles at the end of each sequence can submit the final result. The smart contract checks that the worker assignments are done and then validates if the calling worker is a head worker. The result aggregation of the results can be accomplished in several ways. However, in this paper, we consider accuracy as the main testing criterion for the model. Therefore, we take the mean of the submitted results. The workers that submitted their results are kept track of to prevent double submission,

Algorithm 3 Close Deal Between AI Owner and Buyer

Input: token, price, buyer
remainingAmount amount left to be assigned
withdraws[] array of royalty objects to be withdrawn
share royalty percentage
payment current payment amount to assign
remainingAmount ← price
if *token.childOf* == *address(0)* **then**
    erc721.TransferFrom(token.owner, buyer, token)
    Return: Empty Royalty Array []
else
    share ← tokenRoyalty.value
    accounts, length ← **call**: Algorithm 2(token)
    withdraws ← New Royalty Array[](length + 1)
    for *i* ← 0 **to** *length* **do**
        payment ← remainingAmount × (share × (length - i)) × (decayFactor)
        withdraws[i] ← Royalty(payment, accounts[i])
        remainingAmount ← remainingAmount - payment
    end
    erc721.TranferFrom(token.owner, buyer, token)
    emit: withdraws [],"Withdraw Earnings from Smart Contract"
    return: withdraws []
end

Algorithm 4 Worker Registering for Model Assessment Task

Input: taskNumber
.workerCount counter of current workers
.workers number of required workers
.iterations number of required pipelines
.workersaddr address of the worker
Require: task.workerCount != task.workers × task.iterations && workers[msg.sender].taskNo != taskNumber
task.workersAddr[task.wCount] ← msg.sender
workers.position holds submission order position
workers[msg.sender].taskNo ← taskNumber
workers[msg.sender].position ← 0
task.wCount ← task.wCount + 1
emit: msg.sender, task.wCount - 1

Algorithm 5 Submitting Result of Model Testing

Input: taskNumber, order, result
`// order is the order of the worker in pipeline`
Require:task.wCount == task.workers × task.iterations && workers[msg.sender].position == 0
`// Ensure registration is done, and worker did not submit yet`
Require: order + 1 % task.workers == 0 && task.workersAddr[order] == msg.sender
`// Validate that submitter is an oracle head`
task.result ← task.result + result
task.submitted ← *True*
workers[msg.sender].position ← task.submitted.length
if *task.submitted.length* == *task.iterations* **then**
    task.result ← task.result / task.submitted.length
    emit: task.result, taskNo
    delete task[taskNumber]
end

Algorithm 6 Buyers Bidding on Auction

Input: auctionToken
Require: auction.onGoing && msg.value > auction.highestBid + minimumValue
minimumValue is the smallest increment allowed for new bid
.onGoing is a boolean which checks if auction is ongoing
if *bid.amount != 0 && bid.bidToken* == *auctionToken* **then**
    bid.amount ← bid.amount + msg.value
else
    bid ← newBid(auctionToken, msg.value)
end
auction.highestBid ← msg.value
auction.highestBidder ← msg.sender
if *auction.end - block.timestamp* < *gracePeriod* **then**
    auction.end = block.timestamp + gracePeriod
end
emit: msg.sender, auctionToken, msg.value

and if all head workers submit, the mean gets submitted for the users to employ in their decision-making.

Shifting to the marketplace contract, network users can bid on the asset once an auction starts. Algorithm 6 begins by ensuring that the bid is for a valid ongoing auction and that the bid is higher than the latest bid along with the specified minimum margin. If the user already had an existing bid, it only gets updated. Otherwise, a new bid is issued. Furthermore, if the bids were made during the grace period, they get reset to allow other users a chance to bid.

The auction can end with two options: getting notified by a timing oracle or being stopped by the asset owner. As in Algorithm 7, the auction stops with no checks required if it is the owner. However, if the oracle calls the function, the smart contract checks the current time and penalizes the oracle in transaction history if it is an invalid request. The highest bidder and their bid are then made public on the network, and other bidders can withdraw their bids if they have not already.

Once the auction has ended, the NFT owner initiates a request to transfer their NFT to the auction winner and receive

Algorithm 7 End Auction for the Model NFT

```
Input: auctionToken
Require: oracle.active == true || auction.owner == msg.sender
if auction.owner == msg.sender then
    auction.onGoing ← false
else
    if block.timestamp >= auction.end then
        oracle.successTxs ← oracle.successTxs + 1
        auction.onGoing ← false
    else
        oracle.successTxs ← oracle.successTxs - 1
        revert("Time not up yet!")
    end
end
emit: auction.owner, auctionToken, auction.highestBidder, auction.highestBid
```

Algorithm 8 Tranfer NFT and Assign the Shares for Contributors

```
Input: auctionToken
Require: !auction.onGoing && auction.owner == msg.sender && auction.highestBidder !=address(0)
temp is a temporary array holding addresses with royalties
Royalty [] temp ← call: Algorithm 8(auctionToken, auction.highestBid, auction.highestBidder)
if temp.length == 0 then
    withdrawingAccs ← Royalty(auction.highestBid, auction.highestBidder)
else
    for i ← 0 to temp.length do
        withdrawingAccs[temp[i].recipient] = temp[i]
    end
end
delete auction[auctionToken]
```

the appropriate payment. As such, Algorithm 8 requires that the auction has stopped, the highest bidder is announced, and the owner is the function's caller. Next, Algorithm 3 is called to complete the transfer of the NFT, compute the shares, and assign them to the appropriate addresses. Once the result is returned to Algorithm 8 the addresses are added to an array that temporarily holds them. Each address is associated with the amount they are allowed to withdraw from the marketplace smart contract. Finally, the auction is cleared from the state variables, and users can now participate in other auctions.

B. VALIDATION

We implement the smart contracts using the Solidity programming language. The development environments are Truffle version 5.4.26 and Remix version 0.8.7 [44], [45]. For debugging and deployment, we used Ganache, and we conducted the tests using Truffle's mocha testing framework and chai assertions, which are based on JavaScript. A general test containing several scenarios, assertions, and exception handling is discussed to validate the solution. The general test is divided into 11 tests, each validating a function of the system. In this case, the entity that offers the marketplace service, whether an individual or a Decentralized Autonomous Authority (DOA), is the one that puts both contracts into place.

1) MINTING THE NFTs

Creators and owners of AI assets can mint their assets as NFTs to their associated Ethereum Address. If the asset is derived or inspired by an existing asset, then the user indicates its parent asset. An NFT owner also has to call a function to allow the marketplace contract to be an "Operator", giving the contract the right to operate its NFT on the network. A sample can be seen in Figure 7. **Assert:** 1) Parent of the asset is the predecessor. 2) NFT Owners' approved marketplace contract as operator.

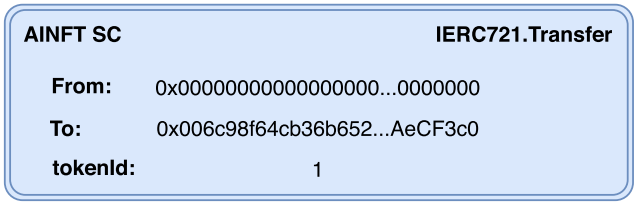

FIGURE 7. Event emitted by the smart contract upon successful minting of NFT.

2) REGISTER AND VERIFY WORKERS

Workers and verifiers can register on the network as participants with their own EA. Verifiers have the task of verifying the attestation of workers and ensuring that workers are running valid TEEs. Consequently, workers must be verified to register and participate in an assessment. Although verifiers submit their attestation results, the worker only gets verified once there is consensus according to the set criteria. For instance, a worker is valid only if the number of verifiers attesting is X more than those that do not, where X is the verifiers' threshold parameter. **Exception Handling:** Verifier attempts to validate the worker twice. **Assert:** Worker verified once consensus requirement is fulfilled.

3) ASSESSING AI MODEL

Any network user interested in validating the claimed model results can initiate a model verification request, such as potential buyers or the owner. Once the request is initiated, the workers can register for the verification task. After workers execute their assigned tasks, worker heads accumulate the final results. Then, the workers' heads are permitted to submit their results to the smart contract while preventing other workers from submitting them. Once all heads

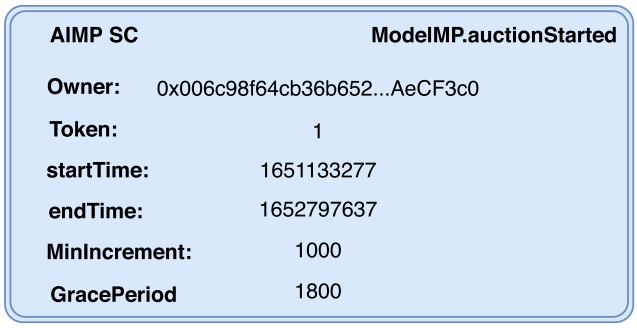

FIGURE 8. Event emitted by the smart contract upon successful auction start.

have submitted, the smart contract publicizes the assessment result, and the model gets verified. Finally, workers can register for new tasks. **Exception Handling:** 1) Worker attempts to register twice for a task. 2) Worker attempts to submit a result, posing as a head of workers. **Assert:** Correct verification of assessed models.

4) AUCTIONING ASSET

Asset owners can auction their assets on the marketplace smart contract by providing the token number and specifying the period, as shown in Figure 8. A time oracle registers to notify the smart contract about the end of the auction period. The auction disallows periods that are not feasible, invalid tokens, or callers other than the owner. Moreover, the owner can preemptively end the auction if satisfied with the highest bid. Once the period ends, the time oracle calls the function to end the auction, and if it is not accurate, the oracle gets penalized. Furthermore, the smart contract prevents any other account from ending the auction, an oracle ending it before time is up, or transfers of tokens before the auction ends. A sample of the event emitted at the end of the auction can be seen in Figure 9. **Exception Handling:** 1) Prevent the auction from starting due to an invalid timeframe. 2) Aborting the start due to an invalid token. 3) Do not start due to an unauthorized account. 4) Prevent the end of the auction due to an unauthorized account (not the owner/oracle). 5) Use Oracle to halt the end before time runs out. 6) Prevent NFT transfers before the auction ends. bf Assert: The account is registered as a time oracle.

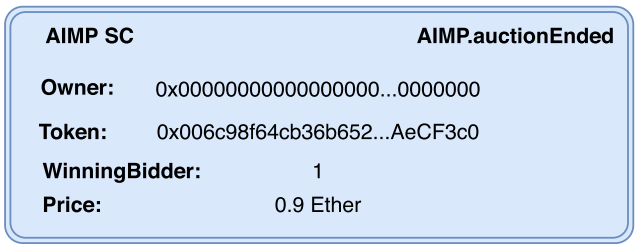

FIGURE 9. Event emitted by the smart contract upon the start of a model verification task.

5) BIDDING AND WITHDRAWING

Users can bid on ongoing auctions identified by the token. Each bid must be higher than the previous one by a set minimum value. Users are not allowed to bid simultaneously in two auctions but can withdraw from auctions before bidding in another. The user cannot bid unless sufficient funds are deposited, which they can withdraw later. If a user is outbid, they can withdraw their funds, but the highest bidder is prevented from doing so. **Exception Handling:** 1) Prevent the user from bidding the same or less than the previous bid. 2) Prevent a non-bidding account from withdrawing funds. 3) Make it impossible for the highest bidder to withdraw funds during the auction.

6) NFT TRANSFER AND PAYMENT

The marketplace smart contract alerts the owner to initiate the token transfer to receive payment for the sold NFT. Each contributor's share in the model is computed based on each sale price's royalty percentage. The transfer only goes through by the owner of the NFT once the auction ends. Only then can contributors and owners collect their assigned shares from the marketplace smart contract through a withdrawal function. Each account is uniquely allowed a specific amount to withdraw. **Exception Handling:** 1) Prevent unauthorized users from transferring data. 2) Prevent the sale of non-auctioned tokens. 3) Prevent payment withdrawals from non-eligible accounts. **Assert: The owner of the NFT is the auction winner**.

VI. DISCUSSION

In this section, we present cost and security analyses of the proposed solution. Also, we discuss the solution's applicability for public AI assets and how it can be generalized for other applications.

A. COST ANALYSIS

The proposed solution is mainly built on the blockchain, which constitutes most of the system's cost. The network is not constrained to Ethereum but rather a public blockchain network. The costs are in terms of gas, a unit that quantifies the computational consumption to execute the operations of a transaction [46]. Gas fees are variable, with each block having a minimum base fee to include a transaction. However, prices are predictable after the London upgrade, and the cost of opcodes is fixed. While gas is consistent and paid in Ether, the price of Ether is not. As such, even if the same transaction is executed twice at two different time points, its price would be different. Table 2 gives an approximate cost summary at the time of testing. This also clarifies the computational requirements of executing a transaction by a user. Every transaction uses 21, 000 gas, and there is a base fee to be included in a block and finally a priority fee which is given to the miners/validators. The rest of the gas is a computational cost for the operations carried out by the EVM (Ethereum Virtual Machine). As such, the

TABLE 2. Smart Contract functions cost analysis.

Task	Caller	Function	Gas Price	Ether	Ether-USD	Matic	Matic-USD
Registrations	Oracle (TEE)	registerWorker	78,300	0.0027	$8.56	0.0029	$0.0043
	Oracle	registerVerifier	44,036	0.0015	$4.82	0.0016	$0.0024
	Worker	registerForTask	71,031	0.0024	$7.77	0.0027	$0.0039
	REP	registerREP	44,025	0.0015	$4.82	0.0016	$0.0024
	Oracle	registerOracle	43,026	0.0015	$4.71	0.0016	$0.0024
Minting	Owner	mint	265,719	0.0090	$29.06	0.0099	$0.0147
	Owner	setApprovalForAll	44,926	0.0015	$4.91	0.0017	$0.0025
Model Assessment	Verifier	verifyWorker	47,269	0.0016	$5.17	0.0018	$0.0026
	User	modelVerificationReq	102,138	0.0035	$11.17	0.0038	$0.0057
	Worker	submitResult	94,473	0.0032	$10.33	0.0035	$0.0052
Auctioning	Owner/Operator	startAuction	118,012	0.0040	$12.91	0.0044	$0.0065
	User	bidOnAuction	93,625	0.0032	$10.24	0.0035	$0.0052
	Bidder	withdrawBid	21,658	0.0007	$2.37	0.0008	$0.0012
	Owner/Oracle	endAuction	66,388	0.0023	$7.26	0.0025	$0.0037
Asset Exchange and Payment	REP/Buyer	sendAsset	28,785	0.0010	$3.15	0.0011	$0.0016
	Owner	transferNFTAndAssignPayments	168,501	0.0057	$18.43	0.0063	$0.0093
	Owners	withdrawPayment	21,795	0.0007	$2.38	0.0008	$0.0012

displayed gas costs represent the computational effort needed to execute the operations of the transaction with the given inputs. Computations have a set cost based on the blockchain specification. In the case of Ethereum, for example, each zero-valued data byte in a transaction is 4 units of gas, while non-zero data is 16 units of gas. While saving to persistent storage using an SSTORE operation costs 20, 000 gas. Such details of the costs of each operation are defined by Ethereum [47].

In the proposed approach, the implementation was designed to reduce costs without compromising functionality. The function to *Mint* is the most costly function (265,719 gas units), which is called from the original ERC-721 smart contract by OZ. Unlike common approaches, the implementation does not require a new smart contract for every user. As such, the user only incurs the function cost of minting the NFT rather than both minting and contract deployment. The highest fee is for the *transferNFT* and *AssignPayments* (168,501 gas units), ensuring that the assignment of payments and transfer of NFT are executed in one transaction. The MP and NFT contracts facilitate the withdrawal of fair payments and ownership transfer, each with its respective task. On the other hand, functions that pertain to auctions, which are called several times, have a relatively low gas cost (approximately 21,800 gas units).

The average gas fee for a function call is 79,630 gas units. As mentioned, the gas price depends on the network's cryptocurrency price. If a PoW consensus network is used, the cost will be around $8.71. However, in a Proof-of-Stake (PoS) consensus-based network such as Polygon (Matic), the cost is $0.0044 [48]. The current high gas fees are due to the scalability of the Ethereum network. Polygon is a side-chain Ethereum network that uses PoS consensus, which Ethereum is implementing in its Beacon chain that is set to merge with the current Mainnet [49]. Therefore, even though the current fees are feasible, the proposed solution has negligible costs using PoS consensus.

B. COMPARISON ANALYSIS

Herein, we compare the proposed solution with the existing solutions based on seven parameters; namely, remuneration, ownership, exchange, verification, provenance, trustless, and model privacy. Compared to the existing solutions [14], [15], [16], [17], [18], [19], [20], [21] as shown in Table 3, it is revealed that only the proposed solution considers all the parameters, as discussed below:

- **Remuneration:** In our solution, participants are paid for their services or resources through a decentralized auction and royalty system.
- **Ownership:** Each participant possesses proof of ownership of a resource, granting them the right to use, possess, and trade the resource. We enforce that using the NFTs on the smart contracts.
- **Exchange:** Participants are able to exchange resources securely and are guaranteed proper compensation. In our solution, this is achieved through SC policies, NFTs, auctions, and trade functions.
- **Verification:** A method is provided for users to verify a resource before purchase to avoid being deceived. Our worker network is utilized to verify a model without compromise of trust or privacy.
- **Provenance:** The information, metadata, and history of change are maintained and available to be audited. Furthermore, it is essential that provenance data is guaranteed to be tamper-free to establish trust in the data. This is ensured through the harmonious off-chain and on-chain traceability enforced by the SCs.
- **Trustless:** Users do not have to trust a third party with their data, and no third party can disrupt a service for the user. We achieve that through the use of TEE as well as the decentralization of tasks by oracles. Finally, the decentralized SC governs their behavior.
- **Model Privacy:** The parameters are treated as IP and are not exposed unless the owner desires to do so. There is no simultaneous collaboration, and models are enhanced after ownership exchange.

TABLE 3. Comparison with the existing works.

Existing Works	Remuneration	Ownership	Exchange	Verification	Provenance	Trustless	Model Privacy
Intelligence via Blockchain [14]	✗	✗	✗	✗	✓	✗	✗
Tracking of Artificial Intelligence Assets [15]	✗	✓	✓	✗	✓	✓	✗
Blockchain Enabled AI Marketplace [16]	✓	✓	✗	✗	✓	✗	✗
SAIaaS [17]	✓	✓	✗	✗	✗	✗	✓
Machine Learning Meets Blockchain [18]	✗	✗	✗	✗	✓	✓	✗
A Marketplace for Trading AI Models [19]	✓	✗	✗	✗	✓	✗	✗
FDPDDL [20]	✗	✗	✗	✓	✓	✓	✓
Towards Industrial Private AI [21]	✗	✗	✗	✗	✗	✓	✓
Proposed approach	✓	✓	✓	✓	✓	✓	✓

TABLE 4. Throughput analysis of Ethereum and Polygon.

Task	Caller	Function	Calls	Total Gas	Ethereum ThroughPut	Polygon Throughput
Registrations	Oracle (TEE)	registerWorker	6	469,800	4.9	31.9
	Oracle	registerVerifier	5	220,180	10.5	68.1
	Worker	registerForTask	12	852,372	2.7	17.6
	REP	registerREP	2	88,050	26.2	170.4
	Oracle	registerOracle	1	43,026	53.6	348.6
Minting	Owner	mint	3	797,158	2.9	18.8
	Owner	setApprovalForAll	3	134,778	17.1	111.3
Asset Assessment	Verifier	verifyWorker	8	378,151	6.1	39.7
	User	modelVerificationReq	2	204,275	11.3	73.4
	Worker	submitResult	4	377,890	6.1	39.7
Auctioning	Owner/Operator	startAuction	1	118,012	19.6	127.1
	User	bidOnAuction	2	187,250	12.3	80.1
	Bidder	withdrawBid	1	21,658	106.6	692.6
	Owner/Oracle	endAuction	1	66,388	34.8	225.9
Asset Exchange and payment	REP/Buyer	sendAsset	3	86,355	26.7	173.7
	Owner	transferNFTAndAssignPayments	1	168,501	13.7	89.0
	Owners	withdrawPayment	3	65,385	35.3	229.4

C. PERFORMANCE ANALYSIS

The performance of the approach is dependent on our implementation and utilization of the blockchain network. Inspecting the Ethereum network, the average time to mine a block is 13 seconds.[2] As for the block size, the maximum allowed size in gas is 30 million.[3] Polygon has a similar block size but a block time of 2 seconds.[4] In the aforementioned networks, both the priority fee and the congestion level play a major role in latency. Providing a decent priority fee and having low network traffic would yield faster transaction time, while lower priority fees and high network traffic would yield latency that is higher than the average. In the former case, the latency could be negligible where a transaction is mined after it is sent if the block is about to be mined, or at worst, it would wait for one block time. Taking the average latency, we simulate the throughput of our implementation as shown in Table 4. The throughput represents the number of actions per second for each function. This applies to the default scenario, which was tested to go through a whole cycle of the system. The number of calls could vary depending on the case, such as wanting to utilize more worker oracles. As the transaction times are much shorter in Polygon than in Ethereum, it is clear that the throughput is higher.

D. SECURITY ANALYSIS

We analyze our prototype using a vulnerability analysis tool to provide a quantitative representation of potential vulnerabilities. The tool utilized in our assessment is slither [50], which is an open source analysis framework providing a suite of vulnerability detectors for smart contracts. There were a total of 13 vulnerabilities detected, as shown in the Table 5, with 7 low-impact, 6 medium-impact, and 0 high-impact vulnerabilities. Manually checking the vulnerabilities uncovered that the vulnerabilities could be considered as false positives. The 6 medium-impact detected vulnerabilities are for uninitialized variables that are recommended to be initialized for readability. The low-impact vulnerabilities mainly consisted of warnings for the use of timestamps in comparisons and a benign re-entrancy vulnerability that is handled. Furthermore, the analyzer provided us with 32 optimization issues and 81 informational issues, which can be included to further enhance the quality of the smart contracts. We investigate the required security requirements for the proposed system, as can be seen below.

TABLE 5. Security analysis matrix of confidence and impact.

Impact \ Confidence	Low	Medium	High
Low	0	4	3
Medium	0	6	0
High	0	0	0

- **Availability**: The proposed approach relies on a decentralized structure at every stage, evident in the blockchain, oracles, PRE, and storage. This approach abolishes the need for trusting a central entity but rather requires trusting the process of the network.

[2] https://etherscan.io/chart/blocktime
[3] https://ethereum.org/en/developers/docs/blocks/
[4] https://polygonscan.com/chart/blocktime

Consequently, tracking AI assets is assured through persistently available data provenance across the many nodes that maintain the blockchain. The same concept applies to IPFS, which seeks to achieve data permanence through a resilient network. In this case, once an asset is uploaded, the owner can not maliciously prevent access to the new owner. As for workers and PRE, the decentralized nature allows some participants to bail on their tasks, and the remaining participants can still carry out the assessment or re-encryption. Furthermore, we utilize PRE to ensure asset availability since the owner might not always be available to share the partitions, so PRE securely transfers the partitions.

- **Data Integrity**: On-chain data integrity is guaranteed by the blockchain's hash chain mechanism, which ensures the immutability of the data once committed to the blockchain, preventing manipulation. Off-chain data integrity is ensured by smart contracts. Every asset is associated with a unique identifier (NFT) on the smart contract by providing the IPNS and IPFS links. Any change to the IPFS entry invalidates the link that exists in the smart contract. Therefore, the user has to update the link, and in doing so, all the users can see the change and choose to utilize the new data or not.
- **IP protection**: Leakage of data or losing its confidentiality deprives the owner of the ability to be compensated for creating the IP. However, the IP needs to be assessed by a potential buyer, which we do by employing a decentralized network of workers. Protecting the IP is done by authenticating workers with valid TEEs and partitioning the model. As such, each partition is tested in a secure enclave, preserving confidentiality. Furthermore, since the model is partitioned, an attacker needs to gain control of all involved workers' TEEs, which is very complicated. Finally, the whole process is tracked on the blockchain to ensure non-repudiation.
- **Trust and Transparency**: Common architectures rely on a central entity such as the cloud or third-party services, which requires trusting the vendor. The vendor also poses as a single point of attack or failure. In this work, we try to shift away from that by consistently decentralizing and distributing power amongst participants. Transactions are transparently displayed for all blockchain participants to facilitate such networks' validity. Moreover, users do not have to trust an entity but rather the consensus process of the network, where only a portion of the participants has to be honest. This trust is reinforced by smart contract incentives for good behavior and penalties to deter bad behavior. Furthermore, transactions are processed by smart contracts, which are automated and transparent, with no entity in control. Smart contracts also enforce ease of access to prevent issues such as escrow, double withdrawal, and impersonation.
- **Collusion**: With the inclusion of several decentralized networks, a consensus method must be in place to achieve a valid result. The most common approach is through majority voting, which is the case for the blockchain and worker networks. To successfully collude and change the outcome in the colluders' favor, the attackers must constitute $N/2 + 1$ of the participants. As for PRE, the scope of attack for colluders is to deny services since the data is encrypted and can not be accessed. We use threshold PRE in particular, where the owner asks for M of N PRE to re-encrypt successfully.

We also investigate the soundness of common attacks on decentralized networks. As these blockchain threats have been documented by the literature, we focus on discussing how the mitigation of the attacks is extended to our decentralized system [51], [52], [53], [54].

Attack 1: Denial of service

The attack vectors could be on the network level or our proposed worker level. For the network, the blockchain is a decentralized network capable of handling DoS attacks since each node is redundantly connected to several nodes. On the worker level, a traditional approach, a worker could refuse to verify a model, and this would halt the functioning of the system. However, as we mentioned, the built-in redundant worker network is resilient up to $N - 1$ lines of workers. Where N is the number of worker lines assembled from K workers, and each line contains M workers, such that $K = M \times N$.

Attack 2: Spoofing attack

The spoofing attack vectors could be on the network level or participant level. On the network level, every participating node has a unique Ethereum address and a unique key pair. The participants themselves in our approach are verified and given roles, to create a role-based access control system on our smart contracts.

Attack 3: Sybil attack

Sybil attack on the blockchain is mitigated by the redundancy of participating nodes. As for the workers, Sybil attacks are prevented through the authentication of the TEE of a worker. By leveraging a Root-of-trust (RoT), each physical entity has a unique proof. This forces every device to register once only.

Attack 4: Eclipse attack

The resiliency of nodes to the eclipse attack increases with the size of the network. In our case, we rely on an already established blockchain, removing concerns about the scale of the number of nodes present. As for workers, this attack would be difficult to carry out as the connections are assigned by the smart contract. As long as the workers are connected to the blockchain they are able to correctly identify the worker to communicate with. The attacker would have to jeopardize the connections of the worker to the blockchain, which would be unfeasible as the number of nodes its connected to increase.

Attack 5: Replay attack

Replay attacks usually occur during hard forks of a blockchain, which is handled by security patches with with the fork. Our concern is the possibility of a replay attack on

the smart contract, which could possibly allow a user to replay a transaction to withdraw more than assigned. We combat this through enforcing a nonce with each upcoming transaction, as well as a flag that would indicate that the transaction is only carried out once. The combination of the nonce, flag and unique address enable the enforcement of a single transaction carried out by a specific address.

E. EXTENSION TO PUBLIC AI ASSETS

This paper focuses on exchanging AI assets, taking into consideration users' concerns. In particular, we examine the case where AI assets are private and maintain that throughout the process. In a public setting, the same approach can be applied to maintain the rights of use and distribution of the owner. This setting is technically simpler since assets are publicly available and no private assessment is required. Specifically, the system relies mainly on the blockchain, smart contracts, and NFTs for traceability and ownership.

1) OWNERSHIP

When assets are publicly available, the owner loses control of how an asset is used. However, one way to regulate asset usage is by having proof of ownership for the asset. Having such persistent proof that is undeniable preserves the rights of the owner as per the regulations that they are subject to as intellectual property. This proof is in the form of an NFT, which acts as an immutable digital receipt. By proving ownership, the owner can license the asset and define terms for distribution. Furthermore, this gives the owner legal protection against piracy and acts as a preventive measure for fraud. The owner is also able to transfer ownership with a clear history of previous ownership of the asset and its changes.

2) PROVENANCE

Assets are published on the blockchain as NFTs, with a path to the asset. Once any change occurs, the existing entry for the asset is no longer valid because the assets have to be stored on content-addressable decentralized storage. If the content changes, then so does the hash or path to the content, ensuring that no changes occur without invalidating the existing hash. The asset is accessible only if it is up-to-date on the blockchain and the distributed ledger. After every change, the owner needs to provide metadata that describes the changes. Since the asset is public, unclear or inaccurate information decreases users' trust and consequently decreases the demand for the asset.

3) REMUNERATION

AI asset owners share their assets for monetary gain and protection. While protection is guaranteed through leveraging ERC-721 smart contracts and NFTs, fair remuneration must be considered in several cases. First, the IPs can be licensed with specific terms that have to be agreed on and executed by the smart contracts. Second, a similar application of terms to distribution rights can be agreed on by both parties. In both cases, this needs to follow the common licensing agreements to have legal liability. Furthermore, the license agreement needs to be included in the metadata of the NFT, which is stored on the IPFS. Third, there is the case where the asset is traded, and the ownership is transferred on the agreed terms. In that case, the original owners get compensated fairly through an auction to sell their assets and gain royalties for subsequent trades. Finally, previous cases can be adjusted so the owners can utilize a royalty-based scheme to benefit from the resale and reuse of their assets.

4) SMART CONTRACTS

Recording the history of ownership on the blockchain serves as an immutable audit trail of information. Smart contracts enable the enforcement of policies related to the AI assets in an automated manner without interference from third parties. Automation of such policies through blockchain and NFTs leads to trusted ownership, trading, and access of AI models. The aforementioned aspects are important factors in a network with publicly accessible assets. However, this is uncommon because assets or intellectual property are typically not shared publicly. As such, we touch on the use of public AI assets as it is embodied in our proposed broader solution, such as provenance and exchange of assets. The proposed approach encompasses the public asset case, where certain parts of the system can be excluded from being utilized. In particular, minting, royalties, auctioning, and exchange are all common components in both models.

F. GENERALIZATION

The proposed approach has been designed for AI assets, specifically focusing on AI models. The approach also permits the use of AI datasets as they go through the same process. The only difference is the validation criteria. Workers in the network can validate and verify the datasets while maintaining confidentiality. This concept can be generalized to any asset that requires private verification. Specifically, the architecture can be used with adjustments mainly made at the worker level, determining the nature of validation. Furthermore, the network participants can form a DOA that specifies the criteria for validating and verifying the used asset. The DOA can be used for resources that adjust models or datasets, such as scripts. On the same system, a script to change the dataset or change a model can be enhanced as in the proposed approach, where the metadata and history of changes are tracked on IPFS.

Expanding the scope further, the essence of the approach is applicable to other use cases that require the provenance of digital assets. The blockchain, NFTs, and decentralized storage preserve the history of changes and ownership of the asset. On the other hand, the subsystems of oracles and PRE enable the assets' verification, validation, and assessment based on the set criteria. The assets differ in assessment criteria and metadata information but not in their format and method. As such, it is crucial to define a standard for the metadata information and assessment of assets to generalize

the solution. A simple use case is the exchange of confidential digital documents divided into secure enclaves, each verifying the document. In turn, the document's privacy and the stakeholders' trust in the document are preserved.

VII. CONCLUSION

In this paper, we showcased how blockchain and NFTs can be leveraged to manage ownership, trading, and access of AI models in a manner that is decentralized, transparent, traceable, reliable, secure, and trustworthy. We discussed the proposed system architecture and its components in detail. We presented sequence diagrams to show the interactions among stakeholders involved in terms of ownership, trading, and access of AI models. We employed the decentralized storage of IPFS and trusted re-encryption oracles to securely fetch, store, and share data related to AI assets. We developed smart contracts and validated them through a detailed use case with exception handling and assertions. We presented algorithms along with their full implementation, testing, and validation details. Furthermore, we conducted a cost analysis to show the affordability of our solution. Our security analysis shows our solution is secure enough against common security threats and attacks. In our discussion, we argued and showed that our proposed solution is valid for private and public AI models, requiring even fewer resources than private AI models. Finally, we discussed that our proposed solution can be easily generalized and utilized for other applications and use cases with minor adjustments and modifications.

REFERENCES

[1] A. K. Kar and A. K. Kushwaha, "Facilitators and barriers of artificial intelligence adoption in business—Insights from opinions using big data analytics," *Inf. Syst. Frontiers*, pp. 1–24, Nov. 2021.

[2] E. Papagiannidis, I. M. Enholm, C. Dremel, P. Mikalef, and J. Krogstie, "Toward AI governance: Identifying best practices and potential barriers and outcomes," *Inf. Syst. Frontiers*, pp. 1–19, Apr. 2022.

[3] M. Cubric, "Drivers, barriers and social considerations for AI adoption in business and management: A tertiary study," *Technol. Soc.*, vol. 62, Aug. 2020, Art. no. 101257.

[4] F. Wang and A. Preininger, "AI in health: State of the art, challenges, and future directions," *Yearbook Med. Informat.*, vol. 28, no. 1, pp. 16–26, Aug. 2019.

[5] L. Cao, "AI in finance: Challenges, techniques, and opportunities," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–38, Apr. 2023.

[6] R. Abduljabbar, H. Dia, S. Liyanage, and S. A. Bagloee, "Applications of artificial intelligence in transport: An overview," *Sustainability*, vol. 11, no. 1, p. 189, 2019. [Online]. Available: https://www.mdpi.com/2071-1050/11/1/189

[7] C. Rikap and B.-. AA Lundvall, "Amazon and Microsoft: Convergence and the emerging AI technology trajectory," in *The Digital Innovation Race*. Cham, Switzerland: Springer, 2021, pp. 91–119.

[8] *Research*. Accessed: Sep. 23, 2021. [Online]. Available: https://ai.google/research

[9] W. Entriken, D. Shirley, J. Evans, and N. Sachs. (Jan. 2018). *Eip-721: Non-Fungible Token Standard*. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721

[10] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," 2021, *arXiv:2105.07447*.

[11] S. M. H. Bamakan, N. Nezhadsistani, O. Bodaghi, and Q. Qu, "Patents and intellectual property assets as non-fungible tokens; key technologies and challenges," *Sci. Rep.*, vol. 12, no. 1, pp. 1–13, 2022.

[12] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," *Hasp@ ISCA*, vol. 10, no. 1, pp. 1–8, 2013.

[13] E. Parliament. (May 2016). *General Data Protection Regulation Document 02016r0679-20160504*. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/

[14] K. Sarpatwar, R. Vaculin, H. Min, G. Su, T. Heath, G. Ganapavarapu, and D. Dillenberger, "Towards enabling trusted artificial intelligence via blockchain," in *Policy-Based Autonomic Data Governance*. Cham, Switzerland: Springer, 2019, pp. 137–153.

[15] P. Lüthi, T. Gagnaux, and M. Gygli, *Distributed Ledger for Provenance Tracking of Artificial Intelligence Assets*. Cham, Switzerland: Springer, 2020, pp. 411–426, doi: 10.1007/978-3-030-42504-3_26.

[16] K. Sarpatwar, V. S. Ganapavarapu, K. Shanmugam, A. Rahman, and R. Vaculin, "Blockchain enabled AI marketplace: The price you pay for trust," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 2857–2866.

[17] N. Six, A. Perrichon-Chrétien, and N. Herbaut, "Saiaas: A blockchain-based solution for secure artificial intelligence as-a-service," in *Proc. Int. Conf. Deep Learn., Big Data Blockchain (Deep-BDB)*, I. Awan, S. Benbernou, M. Younas, and M. Aleksy, Eds. Cham, Switzerland: Springer, 2021, pp. 67–74.

[18] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1178–1187.

[19] L. D. Nguyen, S. R. Pandey, S. Beatriz, A. Broering, and P. Popovski, "A marketplace for trading AI models based on blockchain and incentives for IoT data," 2021, *arXiv:2112.02870*.

[20] L. Lyu, Y. Li, K. Nandakumar, J. Yu, and X. Ma, "How to democratise and protect AI: Fair and differentially private decentralised deep learning," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 2, pp. 1003–1017, Mar./Apr. 2020.

[21] S. A. Khowaja, K. Dev, N. M. F. Qureshi, P. Khuwaja, and L. Foschini, "Toward industrial private AI: A two-tier framework for data and model security," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 76–83, Apr. 2022.

[22] V. Mehri and K. Tutschku, "Flexible privacy and high trust in the next generation internet: The use case of a cloud-based marketplace for AI," in *Proc. SNCNW-Swedish Nat. Comput. Netw. Workshop*, Halmstad, Sweden: Halmstad Univ., 2017, pp. 90–148.

[23] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 468–477.

[24] T. W. Edgar and D. O. Manz, "Science and cyber security," in *Research Methods for Cyber Security*, T. W. Edgar and D. O. Manz, Eds. Waltham, MA, USA: Syngress, 2017, pp. 33–62. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128053492000029

[25] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[26] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, nos. 6–10, p. 71, 2016.

[27] N. Szabo. *Smart Contracts*. Phonetic Sciences. Accessed: Nov. 14, 2020. [Online]. Available: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[28] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Gener. Comput. Syst.*, vol. 105, pp. 475–491, Dec. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X19316280

[29] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer Peer Netw. Appl.*, vol. 14, no. 5, pp. 2901–2925, Sep. 2021.

[30] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," 2018, *arXiv:1806.03287*.

[31] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[32] S. Pinto and N. Santos, "Demystifying arm TrustZone: A comprehensive survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, Nov. 2019.

[33] S. Mofrad, F. Zhang, S. Lu, and W. Shi, "A comparison study of Intel SGX and AMD memory encryption technology," in *Proc. 7th Int. Workshop Hardw. Architectural Support Secur. Privacy*, Jun. 2018, pp. 1–8.

[34] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel software guard extensions: Epid provisioning and attestation services," *White Paper*, vol. 1, nos. 1–10, p. 119, 2016.

[35] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System (White Paper)*. Accessed: Mar. 23, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[36] J. Fairfield, "Tokenized: The law of non-fungible tokens and unique digital property," *Indiana Law J., Forthcoming*, to be published.

[37] *IPFS Powers the Distributed Web*. Accessed: Mar. 3, 2022. [Online]. Available: https://ipfs.io/

[38] N. Nizamuddin, H. R. Hasan, and K. Salah, "IPFS-blockchain-based authenticity of online publications," in *Proc. Int. Conf. Blockchain*. Cham, Switzerland: Springer, 2018,pp. 199–212.

[39] G. Caldarelli, "Understanding the blockchain Oracle problem: A call for action," *Information*, vol. 11, no. 11, p. 509, Oct. 2020.

[40] M. Egorov, D. Nu nez, and M. Wilkison, "Nucypher: A proxy re-encryption network to empower privacy in decentralized systems," *NuCypher whitepaper*, 2018.

[41] *Interplanetary Name System (IPNS)*. Accessed: Mar. 7, 2022. [Online]. Available: https://docs.ipfs.io/concepts/ipns/#example-ipns-setup-with-cli

[42] *Attestation Services for Intel Software Guard Extensions*. Accessed: May 13, 2022. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html

[43] S. Murthy, A. A. Bakar, F. A. Rahim, and R. Ramli, "A comparative study of data anonymization techniques," in *Proc. IEEE IEEE 5th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2019, pp. 306–309.

[44] *Ethereum Ide & Community*. Accessed: Jul. 18, 2022. [Online]. Available: https://remix-project.org/

[45] *Sweet Tools for Smart Contracts*. Accessed: Jul. 30, 2022. [Online]. Available: https://trufflesuite.com/

[46] *Gas and Fees*. Accessed: Aug. 15, 2022. [Online]. Available: https://ethereum.org/en/developers/docs/gas/

[47] G. Wood. (Aug. 2022). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[48] *Bring the World to Ethereum*. Accessed: Aug. 1, 2022. [Online]. Available: https://polygon.technology/

[49] *Ethereum Upgrades (Formerly 'Eth2')*. Accessed: Mar. 23, 2022. [Online]. Available: https://ethereum.org/en/upgrades/

[50] J. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," in *Proc. IEEE/ACM 2nd Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, May 2019, pp. 8–15.

[51] D. Dasgupta, J. M. Shrein, and K. D. Gupta, "A survey of blockchain from security perspective," *J. Banking Financial Technol.*, vol. 3, no. 1, pp. 1–17, Apr. 2019.

[52] K. P. N. Puttaswamy, H. Zheng, and B. Y. Zhao, "Securing structured overlays against identity attacks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 10, pp. 1487–1498, Oct. 2009.

[53] J. R. Douceur, "The Sybil attack," in *Proc. Int. Workshop Peer Peer Syst.* Cham, Switzerland: Springer, 2002, pp. 251–260.

[54] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proc. IEEE INFOCOM 25th IEEE Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–12.