Received 17 June 2022, accepted 14 July 2022, date of publication 19 July 2022, date of current version 26 July 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3192388*

RESEARCH ARTICLE

# Incorporating Registration, Reputation, and Incentivization Into the NFT Ecosystem

**HAYA R. HASAN**[1], **KHALED SALAH**[1], **AMMAR BATTAH**[1], **MOHAMMAD MADINE**[1], **IBRAR YAQOOB**[1], **RAJA JAYARAMAN**[2], **AND MOHAMMED OMAR**[2]

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

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

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

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

**ABSTRACT** Non-Fungible Tokens (NFTs) have recently received immense popularity in the digital art industry. An NFT represents ownership of a unique item that is stored on the blockchain and cannot be changed, replaced, and copied. The current NFT ecosystem falls short in trust features and is prone to illegitimate users, threats, and vulnerabilities. In this paper, we propose a blockchain-based solution for the NFT ecosystem that incorporates registration of the participating actors, involves a decentralized reputation system, provides incentives to its users through rewards, and penalizes misconduct. Our system design is built to ensure trust and credibility in the NFT ecosystem. The proposed solution leverages blockchain's intrinsic security features such as transparency, tamper-proof logs, data integrity, accountability, and non-repudiation. We use the decentralized storage of the InterPlanetary File System (IPFS) to store the NFTs' metadata, whereas their hash is stored on the chain. We present algorithms along with their implementation, testing, and validation details. We demonstrate how our solution, as well as smart contract code, is secure enough against common security threats and attacks. We make our smart contract code publicly available on the GitHub repository.

**INDEX TERMS** NFTs, NFTM, blockchain, reputation, staking, incentives, IPFS, Ethereum, smart contracts, trust, security.

## I. INTRODUCTION

The applications of blockchain have evolved rapidly and vastly to include non-fungible tokens (NFTs). Tokens, which are a representation of digital assets, can be classified as fungible and non-fungible [1], [2]. All fungible tokens are identical, can be replicated and interchanged. However, non-fungible tokens like a piece of art work, digital media, or illustrations are unique assets that hold unique attributes and identifiers on the chain and cannot be divided [3]. NFTs are digital assets implemented on top of the Ethereum blockchain. They are different from the built in Ether crypto currency as they are created using their own standard smart contracts. The first Ethereum-based NFT was developed in 2017 and holds the name CryptoPunks. CryptoPunks are 10,000 unique human art figures that are algorithmically generated, sold, and auctioned in the NFT marketplace (NFTM) [4]. The CryptoPunks were the first of many digital art inspirations, including the well-known CryptoKitties. This is an Ethereum based game where the users collect kitten-shaped figures and trade them amongst each other. The game was developed using Ethereum smart contracts and it pioneered the Ethereum Request for Comment (ERC) 721 non-fungible token standard [5]. Semi-fungible tokens also exist. They combine some of the fungible token features with some of the non-fungible tokens. For example, an asset that is not unique but cannot be interchanged for another category. Such features can exist for theater or entertainment tickets, where the tickets for a certain class can be exchanged for another seat in the same class, but, for example, they

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

cannot be exchanged with a seat ticket from a higher class. Fungible tokens use the ERC-20 standard where as the semi-fungible tokens use the ERC-1155 standard [1].

Blockchain's intrinsic characteristics and features make it an undisputed best fit for NFTs. The distributed ledger has immutable logs, can prove the ownership of the NFTs, holds each user accountable, and all transactions on the chain are transparent and tamper-proof [6]–[8]. For NFTs, blockchain has bridged the gap between the physical and digital worlds. However, NFTMs have suffered from illegitimate buyers and sellers who made NFTMs vulnerable, exploited, and hard to trust. Therefore, the aim of this paper is to eradicate the losses and attacks on NFTMs from the behaviour of disguised users. This paper concentrates on the importance of user registration along with the "know your customer" (KYC) [9] fundamentals, which involves verifying the user's identity on the chain and off chain to ensure that the risks associated with their entry into the marketplace are well-known and sorted. Furthermore, to ensure trust is maintained in the marketplace, a decentralized reputation system is designed into our blockchain-based solution. The reputation of the marketplace, sellers, artists, and buyers is logged on the chain. Users should maintain a good reputable status to expand their privileges in the NFTM. Incentivization of the users using rewards and penalties helps eradicate misbehaviour and encourages cooperation.

We propose a decentralized blockchain-based solution. Our solution uses the Ethereum smart contracts to mint NFTs, trade them in a secure and transparent manner, and hold the programmable logic of the system. It ensures the legitimacy of the NFTs through a registration procedure both on the chain and off the chain. Additionally, our solution addresses the legitimacy issue of the NFTs through the incorporation of the decentralized InterPlanetary File System (IPFS). For each minted NFT, the seller must have first uploaded its metadata to IPFS, and a valid IPFS hash is stored on the chain. We exploit the characteristics of the blockchain, including accountability, non-repudiation, transparency, and tamper-proof immutable data provenance [10]. We maintain a reliable and decentralized reputation system to enforce trust amongst the system users and eradicate fake accounts and illegitimate actions.

### A. RELATED WORKS AND CONTRIBUTIONS

Herein, we examine some of the existing work and research on blockchain and NFTs, as well as blockchain-based decentralized reputation systems.

### B. NFTs

The authors in [1] present a research paper on tokenization and blockchain. In their comprehensive study, they focus on the different categories of tokens, such as fungible, non-fungible, and semi-fungible tokens. In addition to discussing the opportunities of NFTs, they also present the risks and security concerns.

The authors of [11] discuss the NFT state of the art, properties, and security issues as well as challenges. The technical report also includes NFT opportunities such as their importance in the gaming industry, as well as virtual events and the protection of digital assets and collectibles.

The study conducted in [12] concentrates on the market dynamics and security issues of the NFT ecosystem, particularly on the 8 NFTMs. Their research discovered bugs in several marketplaces, as well as malicious trading between users and discrepancies among some of the most expensive sales in the NFT trade.

A conceptual NFT-based patent framework design is proposed by the authors in [4]. The paper describes the solution's various layers, which include storage, authentication, verification, blockchain, and the application layer.The authors focused on utilizing NFTs to protect intellectual properties. Their framework is theoretical and can be improved upon by adding a programmable logic implementation using smart contracts.

The authors in [13] create a decentralized token management infrastructure for game networks. They built a decentralized gaming network based on the ERC-1155 architecture. Their main aim was to eliminate the issues faced with a centralized gaming system through their proposed decentralized architecture.

### C. DECENTRALIZED REPUTATION SYSTEMS

The authors in [14] present a blockhain-based decentralized reputation system. They use their own built-in blockchain to store reputation values. They use a single-dimension reputation system where values are either 0 or 1. In their system, they also include an identity-based encryption system. The full implementation details of the system were not provided, but the authors provided an analysis of their results at the end.

A blockchain-based decentralized reputation system built for public IoT fog nodes is presented in [15]. The paper aims to bridge the gap between fog nodes and Internet of Things (IoT) devices through the proposed trust model. Their implementation is based on the Ethereum blockchain and algorithms programmed using smart contracts. The paper also presents a cost and security analysis of their solution.

The authors in [16] design a blockchain-enabled emission trading scheme (ETS) that incorporates a reputation-based system between sellers and buyers. Their reputation system includes offers as well as priority values and visibility. Their implementation concentrates on the ETS scheme and is specifically customised to be integrated for Industry 4.0.

Although the paper in [17] does not represent a complete blockchain-based reputation system, it represents a trust model that can be used for open, decentralized systems. Their computational model integrates authentication verification, signed certificates, inference rules, and relations.

The authors in [18] present a cross-chain interoperability solution named appXchain. The paper discusses the roles and requirements of interoperating blockchain networks for the healthcare industry. It also analyses the steps needed

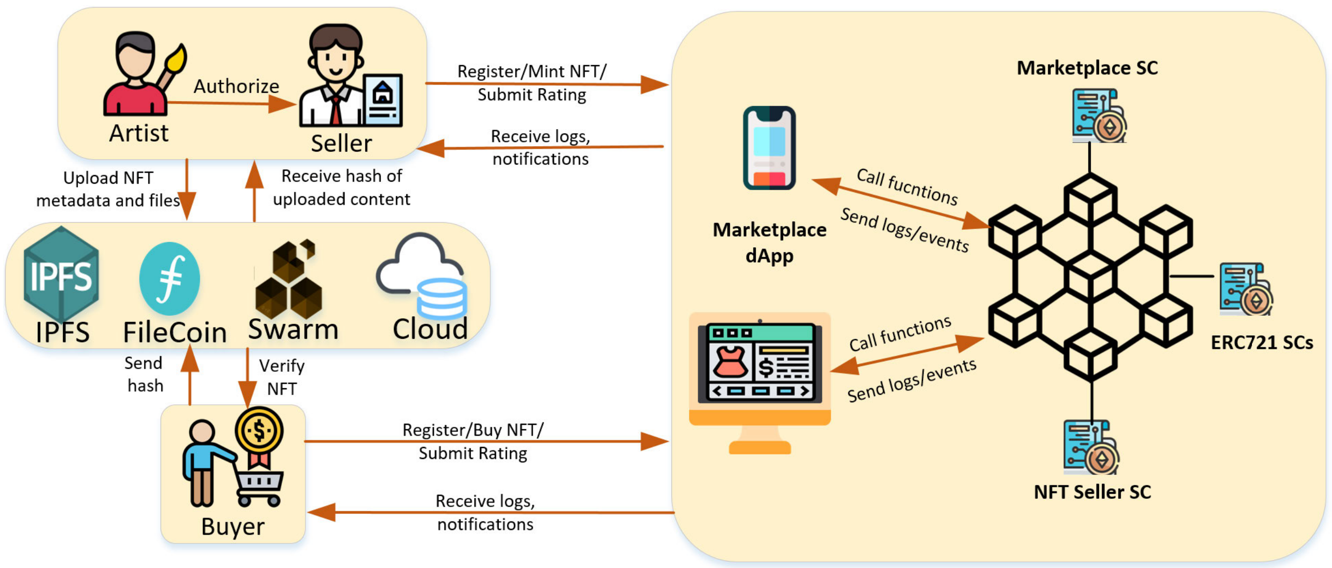


**FIGURE 1.** System diagram of our blockchain-based solution.

to share an electronic medical record (EMR). The authors use an internal and external reputation system. The average reputation score is computed for verifiers, and the count of the ratings is always maintained and used.

PrivBox [19] is a decentralized reputation system designed for online market places. Its cryptographic system depends on zero-knowledge proof to protect the privacy of consumers without the use of third party systems. Ratings are submitted to a bulletin board in encrypted form. Individual feedback values are anonymous and unknown. However, the aggregate statistics of retailers are known to consumers. In their implementation, the authors assume that the marketplace is honest in providing tokens to the consumers, and the Honest-but-Curious (HBC) adversary model is applied to the bulletin board.

In [20], the researchers provide a blockchain-based decentralized reputation system designed for e-commerce. The product information is stored in IPFS and the reputation scores are stored on the chain. Their reputation scores depend on weighted ratings. They have factors specifically designed for the online marketplaces. The design depends on ratings that are both positive and negative, and the factors depend on the transactions between buyers and sellers in online shopping. They also provide a monetary incentive mechanism for their consumers.

RepChain [21] is a reputation system that leverages zero-knowledge and anonymity of its users like [19] but it is blockchain-based and design for E-commerce just like the authors of [20]. In addition to its private ratings, it also allows cross-platform reputation access. The authors evaluate its performance using an Ethereum test network.

The aforementioned solutions all agree on the importance of a decentralized reputation system. A decentralized system is not prone to being a single point of failure, and does not require trusting a third party. In addition, a blockchain-based reputation system leverages blockhain's intrinsic security features and characteristics to achieve transparency, tamper-proof ratings, and immutable records, as well as accountability and non-repudiation. This helps in avoiding fraud and manipulation of the stored reputation scores and ratings. None of the solutions accommodate the NFTs or NFTMs along with a decentralized reputation system. Therefore, our presented solution is a decentralized blockchain-based solution that incorporates IPFS to store the NFTs metadata and a decentralized reputation system that is based on user ratings. The ratings are weighted based on certain computational factors to eliminate fraud, collusion, and unfair ratings. A reward system as well as incentives are part of the design to encourage users to trust each other and to maintain a trustworthy NFTM.

The main contributions of this paper can be summarized as follows:

- We propose a blockchain-based solution that incorporates registration, reputation, rewards, and penalties into the NFT Ecosystem. We integrate the Ethereum blockchain with the decentralized storage of the IPFS to ensure trust and credibility as well as overcome fake and counterfeit NFTs.
- We develop smart contracts that consider many parties, including original artists, NTFMs, buyers, and sellers. We register the participating entities both on-chain and off-chain to maintain KYC records and eliminate illegitimate users and fraud.
- We design a decentralized reputation system that uses weighted ratings to compute the aggregated reputation scores of all participating entities, including the NFTM.

All ratings are weighted on the chain where reputation scores are computed and stored on-chain to ensure non-repudiation.
- We use the weighting factors to compute rewards and incentivize good behavior. Our solution ensures that the re-entry cost to the NFTM is not lower than a poor reputation score. It also expands the visibility of the NFTM auctioned items based on the user's reputation score.
- We present algorithms for registering the participating entities, computing weighted ratings, aggregating reputation scores, rewards, and minting NFT tokens, trading them, and changing their ownership. We make the smart contract code publicly available on GitHub.[1]
- We discuss how our system design and solution are protected against the known security attacks and vulnerabilities of the existing NFTMs. We analyze our smart contract code against security threats and vulnerabilities using the Oyente tool.

The rest of the paper is organized as follows. Section II presents the design details of the proposed blockchain-based solution, followed by the decentralized reputation algorithms and computation details in section III and the implementation details in Section IV including the algorithms. Section V presents the testing details followed by section VI which showcases the security analysis, and open challenges. Section VII concludes the paper.

## II. SYSTEM DESIGN

In this section, we explain the details of the system design. We break the solution into system components and explain their main functions and descriptions. We elaborate on the different processes that take place on the chain and off the chain. This section also helps in showing how the different participating entities, i.e., sellers, buyers, the NFT marketplace (NFTM), and the artists interact with each other through the NFT marketplace and chain interactions. Figure 1 shows how the participating entities, including the seller, buyer, and artist, call functions on the blockchain network through the marketplace. All participating entities must register on the chain to be authorized and accountable for their actions. An artist can authorize a seller to sell their contributions on the network. A seller mints NFTs and lists them on the marketplace for interested buyers. A buyer can verify an NFT using its saved metadata and hash in the off-chain storage. All participants, and even the marketplace, have a reputation score that varies based on the received ratings from the users. A detailed explanation of the different design components is provided in the following subsections.

### A. REGISTRATION OF THE PARTICIPATING ENTITIES

All the participating entities in the system must hold Ethereum Addresses (EAs). In our solution, there are sellers, buyers, and artists. Artists can also be sellers. Buying and selling the NFTs takes place through the marketplace, and the marketplace owner also holds an EA. This ensures that all actions taken by the users are logged on-chain and that all users are held accountable. Registration of the users takes place both on-chain and off-chain. The marketplace SC registers the users on-chain and official documents showing proof of identity are also required off-chain to complete the registration.

The system uses proof-of-stake to ensure that sellers are honest and trustworthy. Therefore, newly registered sellers are all required to deposit a stake amount that is refunded back once they build a good reputation. However, in the future, if the reputation deteriorates and falls below the threshold, then the deposited collateral amount must be paid again to resume activities in the NFT marketplace.

Registration of users also helps in eradicating the wash trading of NFTs. Users might benefit from selling the NFTs to themselves in order to inflate the prices. A report published by Chinalysis shows that NFTs have been sold by the same group of cryptocurrency wallets at least 25 times [22]. However, this dishonest behaviour could be stopped if users were registered both on the chain and off the chain with proof of identity documents. This ensures that each person off the chain is associated with only one single account on the chain.

### B. MARKETPLACE SMART CONTRACT

In our solution, we have two smart contracts that we have created outside of the ERC721 OpenZeppelin [23] smart contracts (SCs). The ERC721 OpenZeppelinThe SCs are used to create ERC721 tokens using their libraries of SCs. The marketplace SC is mainly responsible for registering the participating entities and administering the buying and selling of the NFTs. All interacting and participating entities on-chain must be registered through the marketplace in order to mint or sell NFTs. Registration is only complete if the users also provide proof of their identity off-chain using official attested documents such as passport copies or legal identity documents. Moreover, the marketplace SC saves the EAs of all the NFT seller SCs in a list to ensure that it has a list of all the registered seller SCs. It also recomputes the reputations based on the submitted user ratings after a successful NFT ownership transfer between a buyer and a seller. The marketplace SC can also reward honest buyers and give them incentives to keep the users in an honest environment.

### C. NFT SELLER SMART CONTRACT

The NFT seller SC is owned by a registered seller. Each registered seller will have their own NFT seller SC to mint their own NFTs. The address of the NFT seller SC is then added to the marketplace SC. The NFT seller SC mints NFTs using the NFT metadata such as the URI, price, IPFS hash, and ID. Moreover, this contract also receives calls from the marketplace SC to transfer ownership of a token to a buyer if all conditions are met. The NFT seller SC inherits the functions and attributes of two ERC721 SCs. The inherited functions and attributes are from the ERC721URIStorage

[1]https://github.com/smartcontract694/NFT

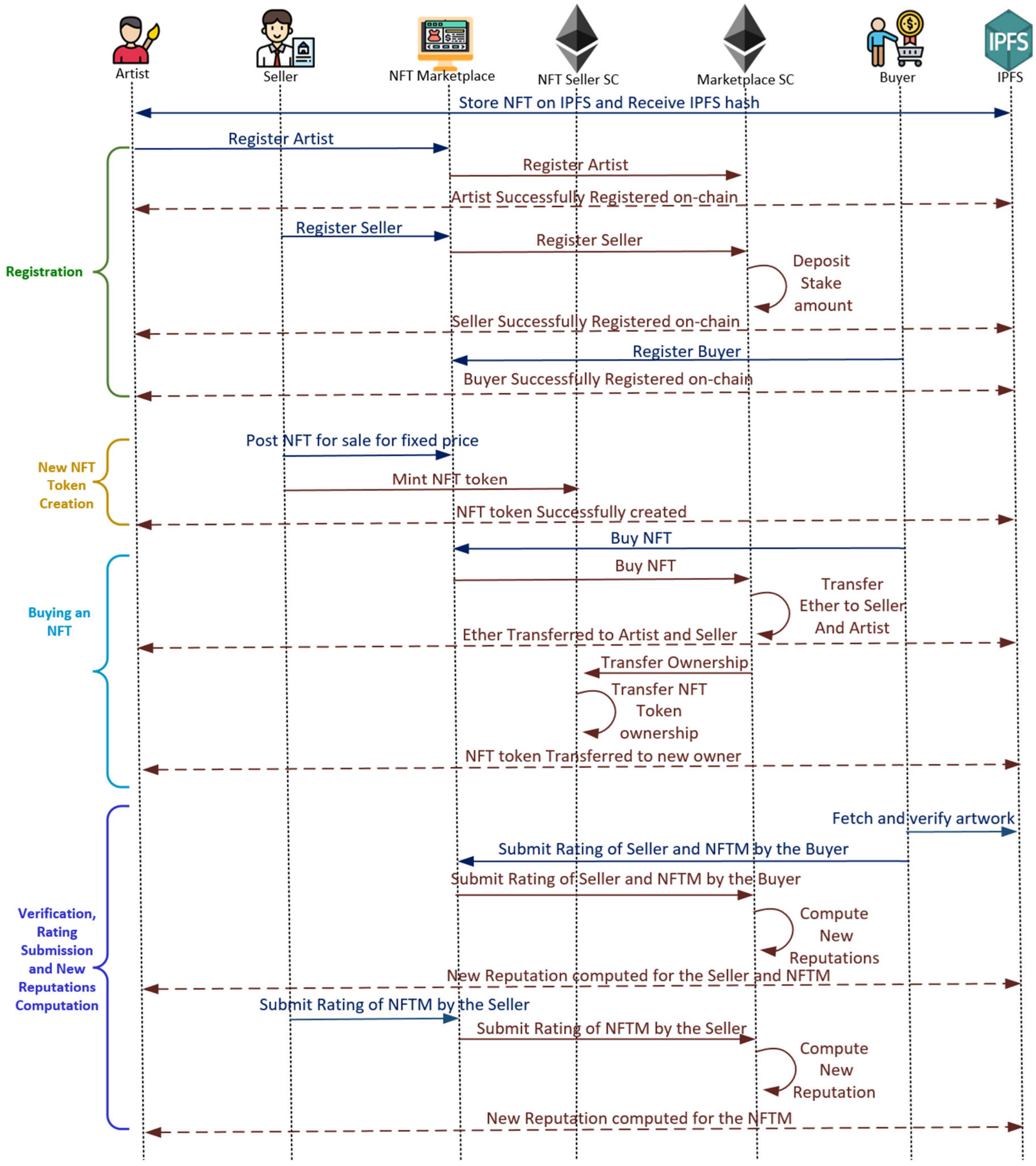

Artist
Seller
NFT Marketplace
NFT Seller SC
Marketplace SC
Buyer
IPFS
Store NFT on IPFS and Receive IPFS hash
Registration
Register Artist
Register Artist
Artist Successfully Registered on-chain
Register Seller
Register Seller
Deposit Stake amount
Seller Successfully Registered on-chain
Register Buyer
Buyer Successfully Registered on-chain
New NFT Token Creation
Post NFT for sale for fixed price
Mint NFT token
NFT token Successfully created
Buying an NFT
Buy NFT
Buy NFT
Transfer Ether to Seller And Artist
Ether Transferred to Artist and Seller
Transfer Ownership
Transfer NFT Token ownership
NFT token Transferred to new owner
Verification, Rating Submission and New Reputations Computation
Fetch and verify artwork
Submit Rating of Seller and NFTM by the Buyer
Submit Rating of Seller and NFTM by the Buyer
Compute New Reputations
New Reputation computed for the Seller and NFTM
Submit Rating of NFTM by the Seller
Submit Rating of NFTM by the Seller
Compute New Reputation
New Reputation computed for the NFTM

**FIGURE 2.** Buying an NFT with a fixed price sequence diagram.

SC, which in turn inherits the functions and attributes of the ERC721 SC.

### *D. IPFS*

Each NFT has metadata stored with it when it is minted. The information stored in the metadata includes a uniform resource identifier (URI), the price, and an identification number (ID). In our solution, the NFT also has a unique hash associated with its ID. The hash is an IPFS hash. Every NFT represents a proof of ownership of an asset, but the information associated with its representation is costly to store on the chain [24]. Hence, it is stored on the decentralized file storage system IPFS, and its unique hash is stored on the chain as part of the metadata. This adds to the authenticity of the NFTs and helps in eliminating fraud.

### *E. BUYING A FIXED PRICE NFT*

An NFT that is posted on the marketplace can be sold either as a fixed price NFT or through an auction. It is the seller's choice to decide on the way the NFT is sold. For a fixed-price

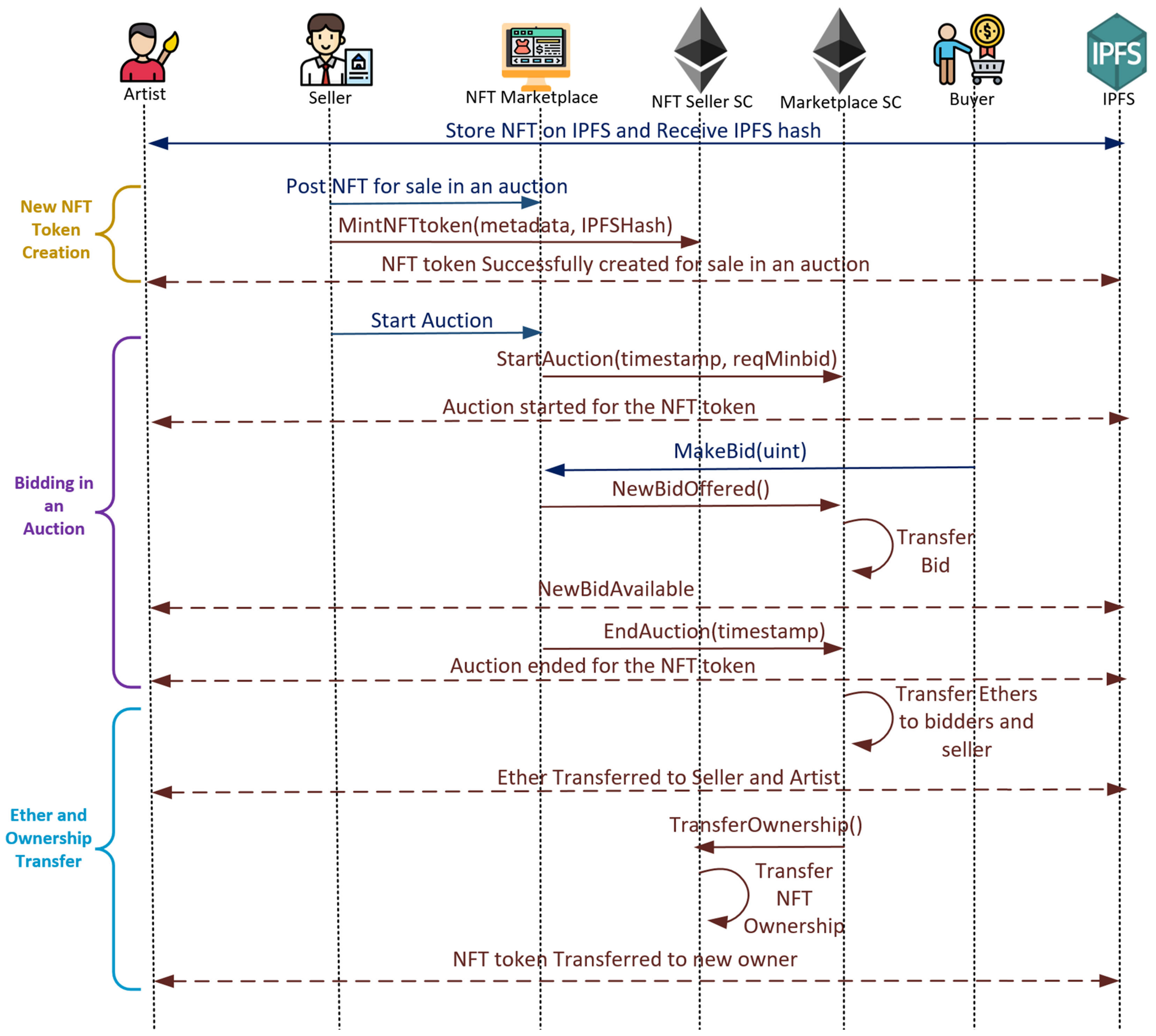

Artist
Seller
NFT Marketplace
NFT Seller SC
Marketplace SC
Buyer
IPFS
Store NFT on IPFS and Receive IPFS hash
New NFT Token Creation
Post NFT for sale in an auction
MintNFTtoken(metadata, IPFSHash)
NFT token Successfully created for sale in an auction
Start Auction
StartAuction(timestamp, reqMinbid)
Auction started for the NFT token
Bidding in an Auction
MakeBid(uint)
NewBidOffered()
Transfer Bid
NewBidAvailable
EndAuction(timestamp)
Auction ended for the NFT token
Transfer Ethers to bidders and seller
Ether and Ownership Transfer
Ether Transferred to Seller and Artist
TransferOwnership()
Transfer NFT Ownership
NFT token Transferred to new owner

**FIGURE 3.** **Buying an NFT through an auction.**

NFT, the process involves an interested buyer requesting to buy the NFT through the marketplace. As shown in figure 2, the buyer requests the NFT from the marketplace. This request is shown as an off-chain interaction using the blue arrows. Then the marketplace connects to the blockchain through the marketplace SC, as can be seen using the brown and red arrows, which indicate an on-chain interaction. In the marketplace, SC transfers the Ether or the price of the NFT to the seller if all conditions are met. For instance, both the buyer and seller must be registered, and the seller's NFT SC address must be available in the saved list of NFT seller SCs that the marketplace SC has. Moreover, the price paid by the buyer must match the price of the NFT asked by the seller. If all conditions are met, the marketplace SC transfers the price to the seller's account and asks the NFT seller SC to transfer the ownership of the NFT to the buyer. The NFT seller SC then transfers the ownership through the ERC721 SC and the new ownership is announced. A buyer can then verify the NFT by using the IPFS hash. Also, the ratings can be submitted by both the buyer and seller. Every rating submitted after a successful transaction leads to a re-computation of the reputation for the intended user. The new reputations are then stored on-chain and are emitted as events to be part of the on-chain logs.

### F. BUYING AN NFT THROUGH AN AUCTION

NFTs can be sold through an auction. The auction is held in the NFT marketplace, and the seller decides on the starting price. The auction is started by the seller, and the buyers start making bids before the end of the auction. The auction is timed using the decentralized application (DApp) timer. The DApp would trigger a call to the marketplace and the auction is ended as can be seen in figure 3. Every time a buyer bids, the bid is transferred to the marketplace SC. Once a bid is made, it cannot be reversed. When the auction ends, the marketplace SC asks for an ownership transfer from the NFT seller SC. The highest bidder is transferred the ownership of the NFT by the seller. The rest of the bidders get a refund of their bids. This method ensures that every bidder is a serious buyer and does not need to deposit collateral to bid. Other auctions, where bidders only make offers without transferring Ethers, risk having bidders change their minds or bid with high amounts and withdraw just to let another bidder with a lower offer win the auction. Hence, this case requires users to pay a collateral before making an offer, and the collateral is not refunded if a bidder acts dishonestly. In addition to lowering their reputation, the collateral deposited helps in eradicating dishonest behaviour by buyers. However, we prefer the method where bidders pay their bid amounts in full when they make their offer. The chosen method ensures that the full bids placed are big enough to discourage dishonesty and eradicate illegitimate behavior.

## III. DECENTRALIZED REPUTATION

Our solution involves a decentralized blockchain based reputation system. Reputation systems enabled by blockchcain offer transparency of the individual users and their on-chain transactions. It also helps in aggregating on-chain data from the immutable distributed ledgers. This helps with trust ratings and reputations and allows the marketplaces to succeed and thrive.

### A. REPUTATION SCORE RANGE

Reputation scores range between a value of 0 to 5. The initial reputation of a new user is 0. Users with a low reputation score that is less than 2.5 pay collateral before executing any purchase transaction on-chain. This puts a burden on users with a low reputation score because they have to delete their accounts and start over with new fake accounts. Our aim is to incentivize the users to be honest and to gradually increase their reputation scores through honest behavior. For instance, this way, a user with a poor reputation score of 0.5 is not incentivized to start over but rather works towards improving their current status. Users are allowed to rate each other as well as the marketplace after every successful purchase transaction. A new rating entered by a user holds a value between 0 and 5 as well. Every new rating entered into the system recomputes the reputation score of the rated entity. It must be noted that all ratings entered as well as reputation scores computed are emitted as events on the chain. Moreover, ratings as well as reputation scores are saved on the chain and increased by a factor of 100 to ease the on-chain calculations since floating points are not supported in Solidity. Solidity is the primary language for developing smart contract codes.

### B. RATINGS FACTORS

Ratings are entered by users after a successful NFT purchase transaction through the marketplace. The ratings entered hold a value between 0.0 and 5.0, and on the chain in the smart contracts, they are increased by a factor of 100. However, ratings are not used as they are when recalculating the reputation score. They are reevaluated using a weight. A new weighted rating is then computed, and the weighted rating is used in the computation of the reputation score. A weighted rating is important because certain factors are ignored if the rating is taken as is and used in computing the reputation. The three main factors that can affect a rating score and are used in finding the weight are:

1) $R_{\text{Cur}}$: Current Reputation Score. It is the aggregated reputation score of a user calculated by the sum of all the previous weighted ratings after every transaction. A user with a poor reputation score will have a lower weight compared to a user with a higher reputation score.
2) $\Delta T_{\text{XY}}$: Timing between two different transactions between the same users. The time between the current transaction ($T_{\text{XYNow}}$) and the last transaction ($T_{\text{XYPrevious}}$) between the same users (X and Y) is important. If the time is short, the weight is less. This factor helps in avoiding scenarios where the buyer and seller might collude to submit good ratings for one another. Equation 1 shows how it is calculated using the timings of both transactions between the users X and Y.

$$\Delta T_{\text{XY}} = T_{\text{XY}_{\text{Now}}} - T_{\text{XY}_{\text{Previous}}} \tag{1}$$

3) $T(N)$: Total number of transactions in the last 6 months. This shows credibility and experience in the marketplace. The higher the number of transactions, the higher the weight of the user. It is also an incentive for the user to continue using the marketplace and to submit ratings. Equation 2 is a generic equation that shows how the sum is calculated to find the total number of transactions in the last N months. In our case, $N$ is equal to 6 and $n$ would take the values from 1 to 6.

$$T(N) = \sum_{n=1}^{N} T(n) \tag{2}$$

### C. WEIGHTED RATINGS

Using the three factors mentioned above, the ratings entered by the users are weighted. First, to compute the weight (W) which will be used to find the weighted ratings, equation 3 is used:

$$W = 0.5R_{\text{Cur}} + 0.35\Delta T_{\text{XY}} + 0.15T(N) \tag{3}$$

Equation 3 is based on the weighted average formula. The weight of a user rating is calculated using the three factors that affect how honest a user rating can be. It depends 50% on the previous reputation of the user, 35% on the time between the transactions with the same user, and 15% on the total number of transactions in the last 6 months. Therefore, the numbers 0.5, 0.35, and 0.15 in equation 3 are used. The weight must be between the values of 0 and 5. Therefore, we have mapped its input values based on the range to values between 0 and 5, as can be seen in table 1. The time difference $\Delta T_{\text{XY}}$ can be less than a few months (M) or can be in years (Y) and

**TABLE 1.** Mapped values used in equation 3.

| $\Delta T_{XY}$ | T(N) | Mapped Value |
|---|---|---|
| $x < 1M$ | < 5 | 0 |
| $x < 3M$ | 5 - 10 | 1 |
| $3M \leq x < 6M$ | 10 - 30 | 2 |
| $6M \leq x < 1Y$ | 30 - 100 | 3 |
| $1Y \leq x < 3Y$ | 100 - 1000 | 4 |
| $x \geq 3Y$ | >= 1000 | 5 |

$T(N)$ shows the total number of transactions that vary and are mapped to a fixed value based on the input range.

The calculated weight is then used to find the weight percentage. The percentage is then multiplied by the entered user rating to find the new weighted rating. Based on the value of the weight, the percentage is given. For instance, a weight of 5 means a 100% and a weight of 0 means 0%. Table 2 shows the different weights values and their corresponding percentages. As the weight (W) increases, the weight percentage (WP) also increases.

**TABLE 2.** The weight values and their corresponding weight percentages.

| Weight(W) | Wight Percentage(WP) |
|---|---|
| 5 | 100% |
| $4 \leq x < 5$ | 90% |
| $3 \leq x < 4$ | 80% |
| $2 \leq x < 3$ | 70% |
| $1 \leq x < 2$ | 60% |
| < 1 | 50% |

The weighted rating ($R_{weighted}$) is then calculated using the new rating ($R_{new}$) entered by the user and the user's corresponding weight percentage as shown in equation 4.

$$R_{\text{weighted}} = R_{\text{new}} * \frac{WP}{100} \tag{4}$$

## D. REPUTATION CALCULATION

The reputation of a user is an insight into their intentions in their past interactions with system users and a possible perception of their typical average behaviour [25]. Therefore, the reputation score is calculated using the average of the aggregated sum of the weighted ratings computed using the equation 4. The weighted ratings are all summed up for any previous interaction with the system users. Then the average is computed. Equation 5 shows how a user's reputation score is calculated. The users are incentivized to rate each other as a higher weight yields from an increase in the number of interactions. Both sellers and buyers can fulfill the needs of each other by rating one another after a purchase interaction.

$$Rep_{\text{score}}(N) = \frac{\sum_{n=1}^{N} R_{\text{weighted}}(n)}{N} \tag{5}$$

## E. REWARDS AND PENALTIES

Users need each other's ratings to build up a good reputation over time. Both buyers and sellers have better advantages with a high reputation. Newly registered sellers as well as sellers with a low reputation score (which is lower than 2.5) pay a stake that is only refunded after building a reputable history of transactions. This proof of stake is submitted by all sellers who are new or whose reputation is lower than 2.5, which helps ensure that sellers maintain their honest behavior towards buyers. Buyers with a poor reputation, on the other hand, have a lower visibility in the marketplace products and auctions.Also, to better incentivise users to be honest, trustworthy, and reliable, sellers can reward buyers after a successful purchase interaction.

A seller might need to be rated by a buyer, and a buyer would love to be rewarded with a rating in return, or by Ether or wei. The reward can be offered by the seller if the buyer agrees to rate the seller. Furthermore, users can also agree to rate each other after a successful purchase interaction. The marketplace online platform can help provide this option for the users and only count the rating of both the users if they both submit a rating. This ensures that ratings are encouraged in the marketplace and enough ratings are available throughout time to compute a good and reliable reputation score.

Buyers with a good reputation score, considered in our system as above 2.5, will have higher visibility of the available auctions and NFTs in the market. They will be able to purchase a wider variety of items compared to a user with a poor reputation score. Also, the seller can offer the buyer a percentage of the NFT purchased as a reward for their honest behavior throughout the purchase. Selling an NFT through an auction to a high bidder who didn't bail before the end of the auction is an incentive for the seller to reward the buyer for their honest behaviour.

On the other hand, a dishonest user will have a poor rating as time passes. This will affect his visibility among the NFTs available in the market. Moreover, it will also affect his weight when rating other users. Another penalty for a poor reputation score that is deteriorating is poor visibility of the available NFTs as well as auctions and not being able to attend auctions from their commencement.

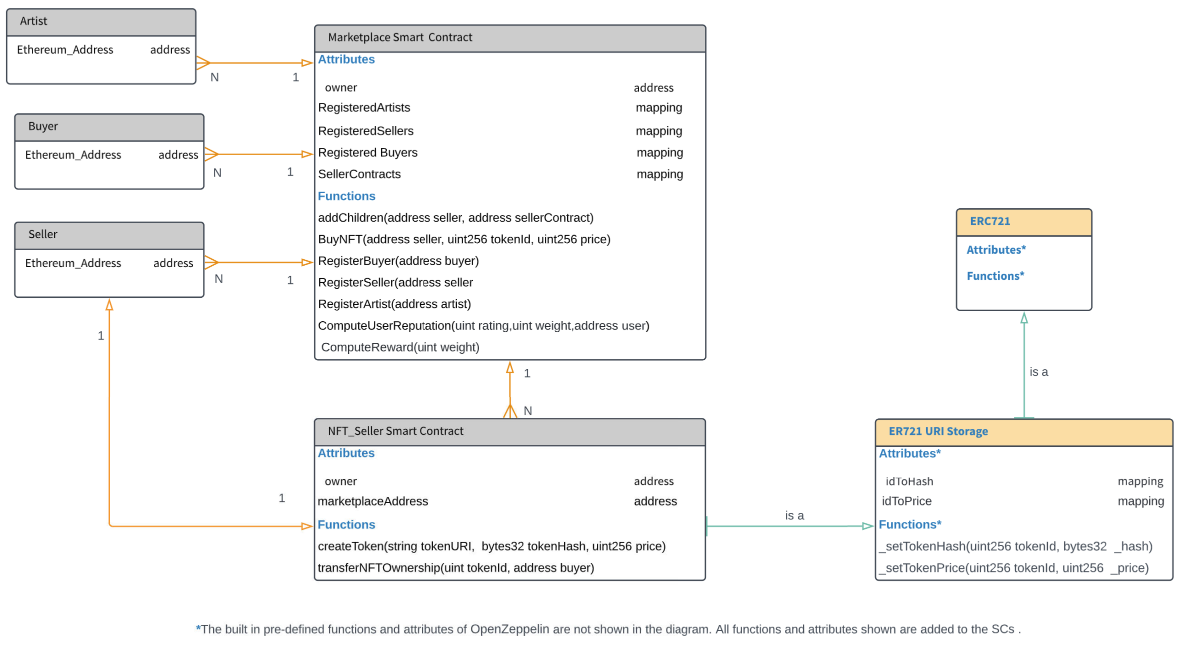
**TABLE 3.** Table mapping the weight with the reward percentage.

| Weight(W) | Reward Percentage(RP) |
|---|---|
| $0 < W < 2$ | 1.5% |
| $2.0 \leq W < 3.5$ | 2.0% |
| $3.5 \leq W \leq 5.0$ | 2.5% |

Rewards submitted to buyers as an incentive for their honest behaviour can be computed using the users' weight. The weight is calculated using the equation 3. Then it is mapped to a percentage, which is used to compute the reward amount from the purchased NFT price. Table 3 shows the weight varying between 0 and 5, mapped to the reward percentage, which varies between 1.5% to 2.5% of the purchased NFT price.

The reward is then calculated using the equation 6. It shows how the reward is calculated from the reward percentage multiplied by the purchased NFT price.

$$Reward = Price_{\text{NFT}} * \frac{RP}{100} \tag{6}$$



Artist
Ethereum_Address address
Buyer
Ethereum_Address address
Seller
Ethereum_Address address
N
1
Marketplace Smart Contract
Attributes
owner address
RegisteredArtists mapping
RegisteredSellers mapping
Registered Buyers mapping
SellerContracts mapping
Functions
addChildren(address seller, address sellerContract)
BuyNFT(address seller, uint256 tokenId, uint256 price)
RegisterBuyer(address buyer)
RegisterSeller(address seller
RegisterArtist(address artist)
ComputeUserReputation(uint rating,uint weight,address user)
ComputeReward(uint weight)
NFT_Seller Smart Contract
Attributes
owner address
marketplaceAddress address
Functions
createToken(string tokenURI, bytes32 tokenHash, uint256 price)
transferNFTOwnership(uint tokenId, address buyer)
ERC721
Attributes*
Functions*
is a
ER721 URI Storage
Attributes*
idToHash mapping
idToPrice mapping
Functions*
_setTokenHash(uint256 tokenId, bytes32 _hash)
_setTokenPrice(uint256 tokenId, uint256 _price)
*The built in pre-defined functions and attributes of OpenZeppelin are not shown in the diagram. All functions and attributes shown are added to the SCs .

**FIGURE 4.** An entity relationship diagram.

## IV. IMPLEMENTATION DETAILS

In this section, we discuss the implementation details and the algorithms used in the smart contract code. The smart contracts are written in solidity using the Remix IDE [26]. Remix is an application that allows the writing, developing, and testing of smart contract code. We have used the OpenZeppelin [23] library to mint NFTs along with their metadata. In our implementation, we have two main smart contracts named the Marketplace Smart Contract (MP SC), and the NFT Seller Smart Contract (SC). The NFT SC connects our code to the OpenZeppelin library by inheriting the ERC721URIStorage. Furthermore, the ERC721URIStorage SC is a descendant of the ERC721 SC.Therefore, our implementation uses functions and attributes from the ERC721 OpenZeppelin library. We have also added functions in the ERC721URIStorage SC as needed to fulfill the needs of our solution. For instance, our solution requires each NFT minted to have a unique IPFS hash and to have its price documented on the chain. Therefore, we have linked the NFT identification value (ID) to a unique hash and to a price value. In the case of selling an NFT in an auction, the saved price acts as the starting price of the NFT. Figure 4 shows the Marketplace SC, and the NFT Seller SC that are created with all their functions and attributes using our code. It also shows the two SCs that the NFT Seller SC inherits from the ERC721 OpenZeppelin library. We have only shown in the diagram the attributes and functions that we have added to those existing ERC721 OpenZeppelin SCs. Inheritance is demonstrated in the diagram using blue arrows. Therefore, the NFT Seller SC *is a* ERC721URIStorage SC which *is a* ERC721 SC. Moreover, several artists, buyers, and sellers are registered on the market place SC. The Marketplace SC is connected to many NFT Seller SCs. But, the NFT Seller SC has only one owner, who is the seller.

Every participating entity must be registered in the marketplace SC. Sellers, artists, and buyers must be registered to execute the function calls. Modifiers are used when needed to restrict access to functions based on the caller's role. Furthermore, every NFT seller's SC Ethereum address must be saved in the market place SC. This allows the marketplace SC to be linked to the registered sellers' SCs and to call functions in their contracts when needed. In the next subsections, the algorithms used in smart contracts will be explained in detail.

### A. MP SC: REGISTRATION OF THE USERS

In the marketplace smart contract, users register so they are allowed to sell and buy using the marketplace. The marketplace SC is created by its owner, who has an Ethereum Address (EA). Therefore, the owner can only register the users using their EAs. Algorithm 1 explains the details of the registration flow that takes place in the marketplace SC. Only the owner is allowed to execute the registration function, otherwise an error is shown. Moreover, there are three mappings used in the smart contract that map the registered users to booleans. Mappings are used instead of arrays to cut costs when searching for a user's EA in the list. Mapping an EA to a boolean makes a registered user get mapped to a *true* while a revoked user to a *false*. If a user is non-existent, then the mapping returns false. Sellers, artists, and buyers all need to register in the marketplace as shown in the algorithm. Each user gets registered based on its category in the respective mapping.

**Algorithm 1** MP SC: Registration of the Users

```
Input : caller, user, MarketPlaceEA, RegisteredBuyers,
        RegisteredSellers, RegisteredArtists, Stakes
1  MarketPlaceEA holds the Ethereum Address of the
     market place
2  caller holds the Ethereum Address of the function caller
3  user holds the Ethereum Address of the user that needs
     to be registered
4  Stakes a mapping that holds true for an EA that
     deposited the stake amount
5  RegisteredSellers is a mapping that stores the EAs of the
     registers sellers
6  RegisteredBuyers is a mapping that stores the EAs of the
     registers buyers
7  RegisteredArtists is a mapping that stores the EAs of the
     registers artists
8  if $caller == MarketPlaceEA$ then
9      if $user == seller$ then
10         if $Stakes[seller] == true$ then
11             RegisteredSellers = true
12         end
13         else
14             Preview an error and return the contract to
                 the previous state.
15         end
16     end
17     if $user == buyer$ then
18         RegisteredBuyers = true
19     end
20     if $user == artist$ then
21         RegisteredArtists = true
22     end
23 end
24 else
25     Preview an error and return the contract to the
         previous state.
26 end
```

Sellers are required to deposit a stake amount at the time of registration, as will be explained in the algorithm 2. The deposited amount will be returned once the seller's reputation exceeds the reputation threshold value. Therefore, before the marketplace registers a seller, the mapping that shows that the stake amount has been deposited for that particular seller is first checked. If the returned value from the *Stakes* mapping is true, then the seller is registered and added to the *RegisteredSellers* mapping.

### B. MP SC: STAKE DEPOSITS

Sellers must first create a proof of stake in order to be registered.Therefore, users are required to pay a deposit before they can register in the marketplace. Algorithm 2 shows the details of how the stake is deposited by a potential seller. Firstly, the marketplace SC would check if the deposited amount is equivalent to the required stake amount. Secondly, the user should only deposit the amount once and should not be already registered as a seller. If all checks are met, then the user's EA is added to the *Stakes* mapping and mapped with a *true*. An event is finally emitted to alert about the new deposit made by the user's EA. This deposited stake amount is needed to increase the re-entrancy value of the marketplace and to incentivize sellers to be honest and build their reputation.

**Algorithm 2** MP SC: Stake Deposits

```
Input : caller, RegisteredSellers, Stakes, stake
1  caller holds the Ethereum Address of the function caller
2  Stakes a mapping that holds true for an EA that
     deposited the stake amount
3  stake the required stake amount to be deposited
4  RegisteredSellers is a mapping that stores the EAs of the
     registers sellers
5  if $Stakes[caller] == false \wedge msg.value ==$
     $stake \wedge RegisteredSellers[caller] == false$ then
6      $Stakes[caller] = true$
7      Emit an event announcing the new deposited stake
         by the caller
8  end
9  else
10     Preview an error and return the contract to the
         previous state.
11 end
```

### C. MP SC: INCLUSION OF THE SELLERS SC IN THE MARKETPLACE SC

Each seller creates an NFT Seller SC that needs to be linked to the marketplace SC. This allows the marketplace SC to control the transfer of Ether between the buyers and sellers. Moreover, it ensures that only registered sellers and buyers interact with each other through the marketplace. Algorithm 3 shows how the EA of the seller's SC is added to the marketplace SC. The marketplace SC owner must be the caller of the function execution, and the seller must be already registered in the marketplace. The seller's NFT SC EA is then mapped to the seller's EA and saved in a mapping in the marketplace SC. An event is then emitted to announce the successful addition of the seller's SC address.

### D. NFT SELLER SC: CREATING NFTs

The seller can create an NFT by minting tokens using the algorithm 4. The seller, who is the owner of the NFT seller SC, is the only one who can create NFTs. Each NFT requires a token URI, a token IPFS hash, and a token price. As can be seen in the algorithm, for each minted NFT, a unique token identification number is created. The algorithm calls the functions from the *ERC*721 library to mint the new token and include its URI in the token URI mapping. Moreover, the token hash as well as the price are also included in the list for the minted tokens. The list maps every ID to its respective hash and price. This is done using the algorithm 10

**Algorithm 3** MP SC: Inclusion of the Sellers SC in the Marketplace SC

```
Input : caller, MarketPlaceEA, RegisteredSellers,
        seller, sellerSC, sellerContracts
1  $MarketPlaceEA$ holds the Ethereum Address of the
     market place
2  $caller$ holds the Ethereum Address of the function caller
3  $user$ holds the Ethereum Address of the user that needs
     to be registered
4  $seller$ holds the Ethereum Address of the seller owning
     the seller SC
5  $sellerSC$ holds the Ethereum Address of the seller SC
6  $RegisteredSellers$ is a mapping that stores the EAs of the
     registers sellers
7  $sellerContracts$ is a mapping that holds the EAs of the
     sellers SCs
8  if
     $caller == MarketPlaceEA \wedge seller \in RegisteredSellers$
     then
9      $sellerContracts[seller] = sellerSC$
10     Emit an event announcing the successful addition of
         the $sellerSC$ for the respective $seller$
11 end
12 else
13     Preview an error and return the contract to the
         previous state.
14 end
```

**Algorithm 4** NFT Seller SC: Creating NFTs

```
Input : caller, tokenURI, tokenHash, seller, tokenPrice,
        marketplaceSC
1  $seller$ holds the Ethereum Address of the seller
2  $caller$ holds the Ethereum Address of the function caller
3  $tokenURI$ is a variable that has the URI of the token
     details
4  $tokenHash$ is a bytes32 that holds the IPFS unique hash
5  $tokenPrice$ is a variable that holds the price of the NFT
6  $marketplaceSC$ holds the EA of the market place SC
7  if $caller == seller$ then
8      $tokenId = ++ tokenIDCounter$
9      Call the $mint(seller, tokenId)$ function of the
         $ERC721$ SC
10     Use the $ER721URIStorage$ SC to map
         $tokenURI \in tokensURIMapping$
11     Execute algorithm 9 to set the token hash and price
12     Approve the $marketplaceSC$ to manage the new NFT
13     Emit an event announcing the minting of the new
         NFT along with the $seller$, $tokenId$, $tokenURI$,
         $tokenHash$ and $tokenPrice$
14 end
15 else
16     Preview an error and return the contract to the
         previous state.
17 end
Output: tokenId
```

which is called inside the algorithm 4. At the end of the execution, the marketplace, in addition to the seller (NFT owner), is approved to manage the NFT. An event is then emitted to announce the creation of a new NFT for the seller.

### E. MP SC: BUY NFT

A registered buyer can buy an NFT through the marketplace SC. The intended seller's EA who is selling the NFT, the NFT's unique ID, as well as the price, are entered as values in the marketplace SC at the time of execution. Algorithm 5 shows the details of the function. The caller must be a registered buyer, and the value entered for the payment must match the price of the NFT. The EA of the seller SC is then found by the marketplace SC using the seller's EA. This is also done to ensure that the seller is a registered one. The seller's SC address is then used to call the transfer ownership function of the seller's SC. The call is made after the buyer is ensured to have the right amount of Ether and the Ether transfer is completed. The ownership is then transferred according to the algorithm 6. An event announcing the change of ownership is emitted at the end.

### F. NFT SELLER SC: TRANSFER OF THE NFT OWNERSHIP

After an interested buyer requests through the marketplace SC to buy an NFT, the transfer of ownership only takes place through a call to the seller's SC. The NFT seller's SC gets a call from the marketplace to transfer the ownership of the NFT to the new buyer. Therefore, as shown in algorithm 6, the caller must be the marketplace SC, and the token ID must exist. Moreover, the marketplace SC must already be approved to manage the NFT along with the owner. If all the conditions are met, the NFT Seller SC calls the transfer function of the ERC721 SC and the ownership is transferred to the new buyer.

### G. MP SC: COMPUTING THE NEW REPUTATION

After every transaction is completed on the marketplace, the reputation is computed for the buyer, seller, and marketplace. This reputation is built in two phases. The first one takes place off-chain, as explained in the previous section, and the second one is on chain. The off-chain process helps in computing a *weight* which is used to better determine the new rating value from the users. In order to calculate the reputation on-chain, the weight is needed, in addition to the user's rating, as well as access to the mappings that hold the users' previous net ratings, number of raters, and the weight percentage. The rating entered on-chain is multiplied by 100 to ease the calculations on-chain and to eliminate issues that might come up from floating points. The weight entered on-chain is a multiple of 100 of the weight computed off-chain. The weight is then mapped based on its range to a value which is used to find the weight percentage. Therefore, as seen in algorithm 7 multiple if-else statements are used to map the values from a respective

**Algorithm 5** MP SC: Buy NFT

```
Input : caller, seller, tokenId, Price, RegisteredBuyers,
        RegisteredSellers
1  caller holds the Ethereum Address of the function caller
2  seller holds the Ethereum Address of the seller
3  tokenId is a variable which holds the unique ID of the
     NFT
4  Price is a variable that holds the NFT price
5  RegisteredBuyers is a mapping that holds the EAs of the
     registered buyers
6  RegisteredSellers is a mapping that holds the EAs of the
     registered sellers
7  if $caller \in RegisteredBuyers \wedge msg.value == price$
     then
8      $sellerContract = RegisteredSellers[seller]$
9      Transfer $price \rightarrow seller$ Call
         TransferNFTOwnership function of the
         sellerContract to execute algorithm 6
10     Emit an event showing that the NFT is transferred to
         a new owner
11 end
12 else
13     Preview an error and return the contract to the
         previous state.
14 end
```

**Algorithm 6** NFT Seller SC: Transfer of the NFT Ownership

```
Input : caller, tokenId, buyer, marketplaceSC, seller,
        tokenOwners, approvedEA
1  seller holds the Ethereum Address of the seller
2  buyer holds the Ethereum Address of the buyer
3  tokenId unique variable that identifies the NFT
4  caller holds the Ethereum Address of the function caller
5  tokenOwners holds the token Ids of the existing tokens
6  marketplaceSC holds the Ethereum Address of the
     market place SC
7  approvedEA hols the EAs that are approved by the seller
     to initiate the transfer of ownership
8  if $caller == marketplaceSC \wedge tokenId ==
     created \wedge marketplaceSC \in
     approvedEA \wedge tokenId \in tokenOwners$ then
9      Call the transferFrom function of the $ERC721$ SC
         using the paramters $seller, buyer, tokenId$
10 end
11 else
12     Preview an error and return the contract to the
         previous state.
13 end
```

range to a particular value. This value is then used inside a mapping called *WeightPercentage* to find the percentage of the user's weight. Therefore, a weight of 50 corresponds to a weight percentage of 10% and a weight of 500 corresponds to a weight percentage of 100% respectively. This weight percentage is then multiplied by the user's rating to come up with the new weighted rating. The weighted rating is then added to the accumulated reputation for each user. Then the number of raters for that particular user is incremented by one and then saved. The new weighted rating, the new accumulated reputation, as well as the number of raters are all emitted as an event. This event is saved in the logs and can be used off-chain to compute the exact reputation of the user. The exact reputation would be the emitted reputation value divided by 10,000 and then divided by the total number of raters, as presented in equation 7.

$$NewReputation = \frac{(EmittedReputation/10000)}{TotalNumberOfRaters} \tag{7}$$

### H. MP SC: REWARD COMPUTATION

A buyer is rewarded for being honest, especially after an auction. Therefore, the marketplace SC can be used by a seller to reward a buyer. The reward is computed based on a weight calculated off-chain. The weight is then mapped to a value. As seen in algorithm 8, multiple if-else statements are used to map a range to a specific value. If the weight, which is multiplied by 100 on-chain, is between 200 and 350, the returned reward value is 20. This indicates that it is 2% of the NFT selling price. This algorithm returns the reward percentage and is an internal function that can only be used inside the marketplace SC.

### I. MP SC: REWARDING THE BUYER

In order to reward an honest buyer after a successful transaction, the seller executes a function call in the marketplace SC. The seller must be registered, as can be seen in the algorithm 9. The reward percentage is then computed using the algorithm 8. The reward is then calculated using the NFT price and the equation 8.

$$Reward = \frac{RewardPercentage * Price}{1000} \tag{8}$$

Using the algorithm 8, the function checks if the value of Ether or wei entered by the seller matches the reward value in order to continue with the execution. The reward is then transferred to the buyer from the seller. An event is emitted at the end to say that the buyer was paid by the seller.

### J. ERC721URIStorage SC: SETTING THE NFT PRICE AND HASH

The NFT Seller SC inherits the functions from the ERC721URIStorage which inherits from the ERC721. Those functions are a crucial part of the NFT Seller SC and allow the contract to mint NFTs and store their metadata such as URI, token ID, token price, and hash. The token price and token hash are not built-in in the ERC721 metadata. Therefore, we have added the functions to set the token IPFS hash and set the token price in the ERC721URIStorage. As seen in algorithm 10, the token must exist in the list of token IDs in order to map the token ID to its hash and price. Two mappings

**Algorithm 7** MP SC: Computing the New Reputation

```
Input : caller, rating, weight, user, marketplace,
        WeightPercentage, UserRatings,
        UserNumberofRaters
1  $rating$ a variable that holds new rating by the user
2  $weight$ a variable that holds the calculated user weight
3  $WeightPercentage$ is a mapping that maps the rating
     weight with its percentage
4  if $caller == marketplace$ then
5      if $weight >= 0 \wedge weight < 50$ then
6          $weight = 0$
7      end
8      if $weight >= 50 \wedge weight < 100$ then
9          $weight = 50$
10     end
11     if $weight >= 100 \wedge weight < 150$ then
12         $weight = 100$
13     end
14     if $weight >= 150 \wedge weight < 200$ then
15         $weight = 150$
16     end
17     if $weight >= 200 \wedge weight < 250$ then
18         $weight = 200$
19     end
20     if $weight >= 250 \wedge weight < 300$ then
21         $weight = 250$
22     end
23     if $weight >= 300 \wedge weight < 350$ then
24         $weight = 300$
25     end
26     if $weight >= 350 \wedge weight < 400$ then
27         $weight = 350$
28     end
29     if $weight >= 400 \wedge weight < 450$ then
30         $weight = 400$
31     end
32     if $weight >= 450 \wedge weight < 500$ then
33         $weight = 450$
34     end
35     else
36         $weight = 500$
37     end
38     $newRating = rating * WeightPercentage[weight]$
39     $newReputation = newRating + UserRatings[user]$
40     $UserRatings[user] = newReputation$
41     $newUserCount = ++ UserNumberofRaters[user]$
42     Announce the new values
43 end
44 else
45     Error. Revoke state.
46 end
```

**Algorithm 8** MP SC: Reward Computation

```
Input : weight
1  $weight$ is a variable that holds the calculated user weight
2  if $weight >= 0$ $weight <= 500$ then
3      if $weight == 0$ then
4          Emit an event announcing reward percentage
5          return 0
6      end
7      if $weight > 0 \wedge weight < 200$ then
8          Emit an event announcing reward percentage
9          return 15
10     end
11     if $weight > 200 \wedge weight < 350$ then
12         Emit an event announcing reward percentage
13         return 20
14     end
15     if $weight >= weight <= 500$ then
16         Emit an event announcing reward percentage
17         return 25
18     end
19 end
20 else
21     Revert contract state and show an error
22 end
Output: rewardWeight
```

are used to set the values, respectively. If the token ID does not exist, an error message is returned and the state is reverted.

## V. TESTING AND VALIDATION

The smart contracts developed for the implementation are written in Solidity using the Remix IDE [26]. The tool allows the programmers to write, test, and debug the code if needed. The functions are tested using the correct input to check the correct expected execution results. Furthermore, functions are also tested based on their restrictions, such as modifiers or requirements. This section presents testing scenarios as well as results. The results show the logs of the successful executions.

In our testing scenarios, the marketplace SC owner holds $0 \times 5B38Da6a701c568545dCfcB03FcB875f56beddC4$ as Ethereum address (EA), the NFT seller SC owner holds $0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2$ as EA, and the registered buyer and new owner holds $0 \times 4B20993Bc481177ec7E8f571ceCaE8A9e22C02db$ as EA.

### A. STAKE DEPOSIT AND SELLER REGISTRATION

The marketplace SC is created by the marketplace owner. The marketplace SC is where the registration takes place. A prerequisite to registering sellers is proof of stake. The seller used the *DepositStake* function call to deposit 5 Ether as can be seen in the logs of figure 5. The user's EA is also shown in the logs, which indicates the EA of the seller. After a successful stake deposit, the seller is then registered with the marketplace

### B. INCLUSION OF THE SELLER SC IN THE MP SC

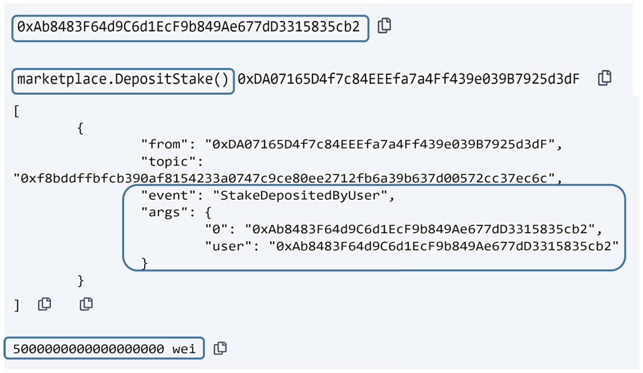
Every seller has an NFT seller SC. This contract should be added to the marketplace SC for all registered sellers. This is

**Algorithm 9** MP SC: Rewarding the Buyer

```
Input : caller, seller, buyer, weight, price, tokenId,
        RegisteredSellers
1  caller holds the Ethereum Address of the function caller
2  seller holds the Ethereum Address of the seller
3  buyer holds the EA of the buyer
4  RegisteredSellers is a mapping that saves all the EAs of
     the registered sellers
5  weight is a variable that holds the calculated user weight
6  price is a variable that holds the price of the NFT
7  tokenId is variable that holds the uniquely identifies the
     NFT
8  if $caller == seller \wedge seller \in RegisteredSellers$ then
9      rewardPercentage = Execute algorithm 7
10     $reward = (rewardPercentage * price)/1000$
11     if $msg.value == reward$ then
12         Transfer reward → buyer
13         Emit an event announcing that the buyer is
             rewarded successfully using buyer, seller and
             reward
14     end
15 end
16 else
17     Preview an error and return the contract to the
         previous state.
18 end
```

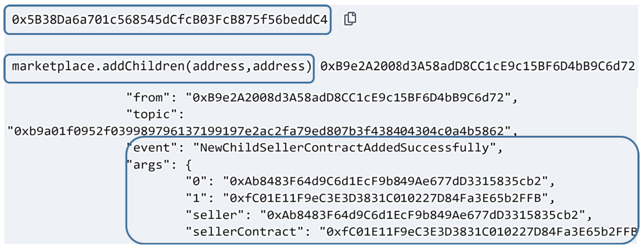
**Algorithm 10** ERC721URIStorage SC: Setting the NFT Price and Hash

```
Input : tokenId, hash, price, tokenIDs, idToHash,
        idToPrice
1  tokenId is a variable that uniquelt identifies the NFT
2  hash holds the hash of the NFT
3  price is a variable that has the price of the NFT
4  tokenIDs contains a list of all the existing already minted
     token IDs
5  idToHash is a mapping between the token ID and hash
6  idToPrice is a mapping between the token ID and price
7  if $tokenId \in tokenIDs$ then
8      $idToHash[tokenId] = hash$
9      $idToPrice[tokenId] = price$
10 end
11 else
12     Alert with an error message stating
         ERC721URIStorage : URIsetofnonexistenttoken
13 end
```



```
0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

marketplace.DepositStake() 0xDA07165D4f7c84EEEfa7a4Ff439e039B7925d3dF
[
        {
                "from": "0xDA07165D4f7c84EEEfa7a4Ff439e039B7925d3dF",
                "topic":
"0xf8bddffbfcb390af8154233a0747c9ce80ee2712fb6a39b637d00572cc37ec6c",
                "event": "StakeDepositedByUser",
                "args": {
                        "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "user": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
                }
        }
]

5000000000000000000 wei
```

**FIGURE 5.** Logs showing a successful stake deposit by the seller.



```
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

marketplace.addChildren(address,address) 0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72
                "from": "0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72",
                "topic":
"0xb9a01f0952f0399897961371991 97e2ac2fa79ed807b3f438404304c0a4b5862",
                "event": "NewChildSellerContractAddedSuccessfully",
                "args": {
                        "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "1": "0xfC01E11F9eC3E3D3831C010227D84Fa3E65b2FFB",
                        "seller": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "sellerContract": "0xfC01E11F9eC3E3D3831C010227D84Fa3E65b2FFB
```

**FIGURE 6.** Logs of the successful addition of the NFT seller SC address to the MP SC.

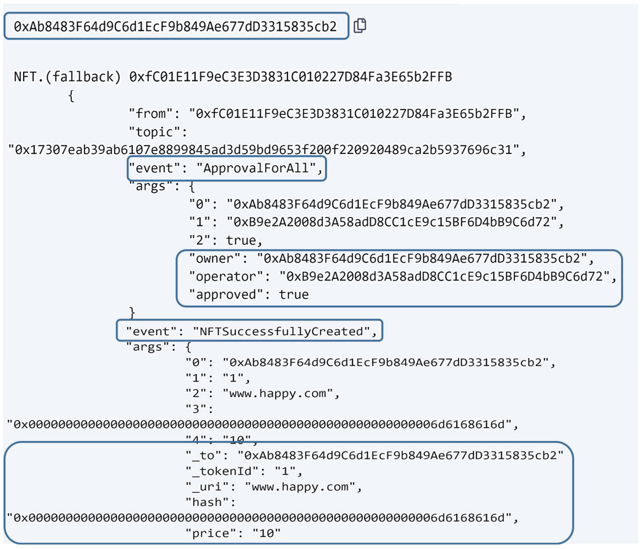
important to enable sellers to sell their NFTs through the marketplace. The *AddChildren* function can only be called by the marketplace. Therefore, the *from* in the logs in figure 6 shows the EA of the marketplace. The function takes the seller's EA as input as well as the NFT seller's SC address. The seller must be registered with the marketplace to be added to the list. An event indicating the successful mapping of the seller and the seller's SC address is emitted at the end.

## C. MINTING NFTs

Sellers can mint NFTs through the NFT Seller SC. Figure 7 shows the events generated from successfully minting an NFT. When a seller mints an NFT, the seller also allows the marketplace SC to be approved as an *operator*, through a function call to the ERC721 predefined functions. The event *ApprovalForAll* shows how the marketplace SC EA $xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72$ is approved with a *true* as an operator. This is then followed by an event *NFTSuccessfullyCreated* showing the successful creation of the NFT for the seller. As can be seen in the figure, the NFT's metadata such as owner, token ID, URI, IPFS hash, and price are all listed as part of the logs.

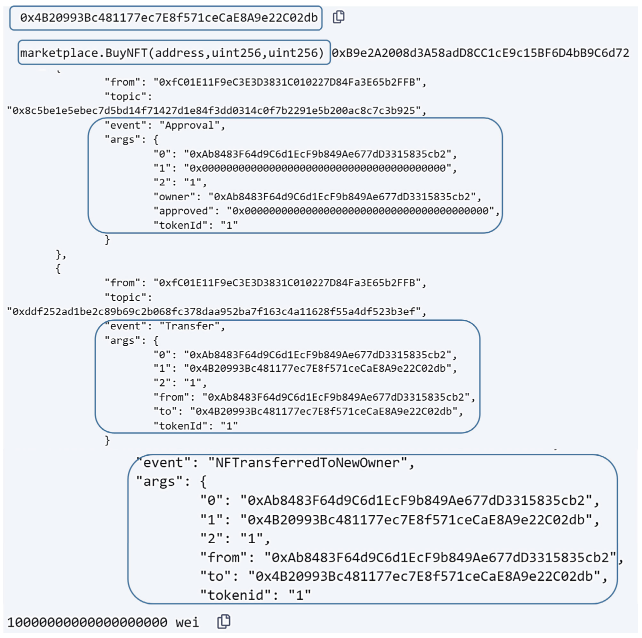
## D. BUYING AN NFT

A registered buyer can only attempt to buy an NFT through the marketplace SC. Using the function *BuyNFT*, the seller's EA, NFT ID and price are entered by the buyer. The marketplace uses the seller's EA and checks if it's mapped to an existing seller's NFT SC. The marketplace SC then checks the Ether deposited by the buyer and ensures that it matches the price of the requested NFT. If all checks are met, then the marketplace SC calls a function in the NFT seller SC to transfer the ownership. The NFT seller SC calls the predefined *transfer* function of the ERC721 and the approval is first checked, as can be seen in the logs of figure 8. The event *Approval* is then followed by the event *Transfer* to show the



```
0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

 NFT.(fallback) 0xfC01E11F9eC3E3D3831C010227D84Fa3E65b2FFB
        {
                "from": "0xfC01E11F9eC3E3D3831C010227D84Fa3E65b2FFB",
                "topic":
"0x17307eab39ab6107e8899845ad3d59bd9653f200f220920489ca2b5937696c31",
                "event": "ApprovalForAll",
                "args": {
                        "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "1": "0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72",
                        "2": true,
                        "owner": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "operator": "0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72",
                        "approved": true
                }
               "event": "NFTSuccessfullyCreated",
               "args": {
                       "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                       "1": "1",
                       "2": "www.happy.com",
                       "3":
"0x0000000000000000000000000000000000000000000000000000006d6168616d",
                       "4": "10",
                       "_to": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
                       "_tokenId": "1",
                       "_uri": "www.happy.com",
                       "hash":
"0x0000000000000000000000000000000000000000000000000000006d6168616d",
                       "price": "10"
```

**FIGURE 7.** Logs showing a registered seller minting an NFT.



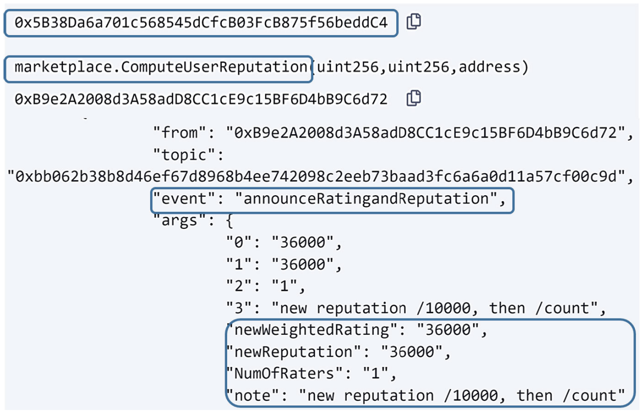
```
 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

 marketplace.BuyNFT(address,uint256,uint256) 0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72
                "from": "0xfC01E11F9eC3E3D3831C010227D84Fa3E65b2FFB",
                "topic":
"0x8c5be1e5ebec7d5bd14f71427d1e84f3dd0314c0f7b2291e5b200ac8c7c3b925",
                "event": "Approval",
                "args": {
                        "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "1": "0x0000000000000000000000000000000000000000",
                        "2": "1",
                        "owner": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "approved": "0x0000000000000000000000000000000000000000",
                        "tokenId": "1"
                }
        },
        {
                "from": "0xfC01E11F9eC3E3D3831C010227D84Fa3E65b2FFB",
                "topic":
"0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
                "event": "Transfer",
                "args": {
                        "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "1": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                        "2": "1",
                        "from": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "to": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                        "tokenId": "1"
                }
            "event": "NFTransferredToNewOwner",
            "args": {
                    "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                    "1": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                    "2": "1",
                    "from": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                    "to": "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db",
                    "tokenid": "1"
1000000000000000000 wei
```

**FIGURE 8.** Logs of successfully buying an NFT by a registered buyer.

transfer of ownership from the seller to the buyer's address. The final event, *NFTransferredToNewOwner* is emitted in the marketplace SC and shows the successful transfer of the NFT ownership.

### E. REPUTATION COMPUTATION

The reputation of the users is computed after a successful transaction between a buyer and a seller. The marketplace receives the readings from the seller and buyer and, based on the calculated weight of the user, the reputation is calculated. The logs in figure 9 show the reputation calculation for the registered buyer using the *ComputeUserReputation* function



```
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

marketplace.ComputeUserReputation(uint256,uint256,address)
0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72
                "from": "0xB9e2A2008d3A58adD8CC1cE9c15BF6D4bB9C6d72",
                "topic":
"0xbb062b38b8d46ef67d8968b4ee742098c2eeb73baad3fc6a6a0d11a57cf00c9d",
                "event": "announceRatingandReputation",
                "args": {
                        "0": "36000",
                        "1": "36000",
                        "2": "1",
                        "3": "new reputation /10000, then /count",
                        "newWeightedRating": "36000",
                        "newReputation": "36000",
                        "NumOfRaters": "1",
                        "note": "new reputation /10000, then /count"
```

**FIGURE 9.** Logs showing the computation of the reputation for the buyer.

in the marketplace SC. The rating, weight, and user's EA are entered as inputs, and the new reputation is computed that is 3.6, as can be seen in the logs. The reward functions were also tested successfully. As expected, the computed reward was correctly computed based on the user's weight and the NFT price.

## VI. DISCUSSION

In this section, we evaluate the security aspects of our solution and show how it is different from the other existing solutions. Also, we outline the limitations of our solution in terms of open challenges.

### A. SECURITY ANALYSIS

Herein, we analyze our system design against the known security threats and demonstrate how it withstands some of the most popular attacks. Also, we analyze our smart contract code using the Oyenete tool and present the security analysis report as part of the result.

#### 1) RESILIENCY AGAINST POPULAR ATTACKS

##### *a: COLLUSION ATTACKS*

A collusion attack is a common attack that takes place when two or more peers cooperate together to take down the reputation of a certain targeted peer [12]. It can also take place in the opposite way, where it can be used to increase the reputation of a specific seller or user. In our design, we ensure that the time between transactions between two users is part of the weighting factors that affect how legitimate a submitted rating is. Therefore, ratings submitted for a transaction between the same set of users within a few minutes of each other are not counted multiple times and have no effect on the total reputation score. Table 1 shows how the time between the last two transactions of the same user affects the weight of the rating.

##### *b: BAD-MOUTHING ATTACK*

It is very likely that a user might feel tempted to submit an unfair rating to another user [27]. Therefore, ratings are all weighted before they are used to compute the aggregated reputation score. A user's total number of transactions in the market place as well as their reputation score affect how reputable the rating submitted is. Therefore, users who have just joined or who have low reputation scores have low weights in their submitted ratings until they prove themselves to be reputable. The user's reputation score always shows how well other people have rated them.

#### *c: SYBIL ATTACK*

This attack takes place when the system allows a single user to create multiple accounts with multiple identities. Hence, several identities on the system are associated with a single person physically [12], [27]. We address this problem by registering all users on the chain and off the chain. Moreover, as a countermeasure in the KYC procedure, all identities on the chain are associated with a unique real-world identity that must be presented off the chain.

#### *d: ECLIPSE ATTACK*

An NFTM cannot alter, or manipulate the submitted ratings or any computed reputation score on the chain. However, the NFTM can block the submitted ratings from the blockchain. This is called the ''eclipse attack,'' where a false view of the blockhain is viewed by the users through the NFTM. The transaction goes from the user's wallet to the NFTM. The NFTM cannot edit the transaction but can stop, hold, or delay it. This is a general problem that gateways and wallets suffer from. However, our system also computes a reputation score for the marketplace as well. If a user's rating is not submitted on the chain, then the user can rate the marketplace poorly, which will lower its score. Moreover, the marketplace would need a high number of raters to increase its credibility and enhance its reputation. Another possible solution, which can also enhance our design further, is to incorporate multiple gateways and NFTMs that can receive the ratings. This would ensure that the rating is submitted on time on the chain. However, it would compromise the size and number of the SCs involved as well as the cost.

#### *e: RE-ENTRY ATTACK*

The re-entry attack happens when the cost of entering the system is low and the reputation score of a new user is higher than the reputation score of a poor user. A malicious user can act poorly and lower the reputation score a lot. Hence, this might incentivize the user to leave the current account and create a new one to have a fresh start with a better reputation [12], [27]. Creating a new account in our system is associated with real identities and off-chain registration. Also, it involves a proof of stake, where a new user pays a certain, agreed-upon amount as collateral. The collateral is maintained until the user proves to be reputable. Hence, it is not too simple to start over with a new account. Moreover, a new user's reputation is considered on the NFTM to be a low one, but the number of transactions in the system as well as the joining date are visible to others to indicate that the low reputation score is due to being new on the system. Hence, a user with a low reputation score would not gain a higher or better reputation score when re-entering the system as a new user. Good behavior must still be adhered to to increase the score and gain better privileges in the marketplace.

#### *f: BID SHIELDING*

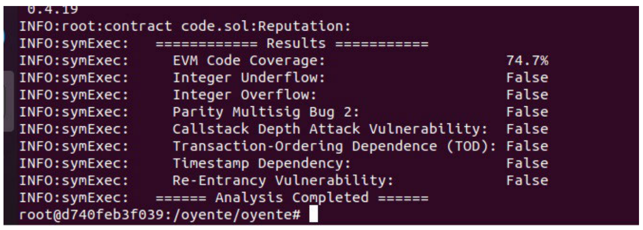
It is a case of collusion where a malicious bidder bids on an NFT with a low bid and another colluding user bids a very high bid. The high bid discourages legitimate users from placing their bids. However, just before the end of the auction, the highest bidder withdraws from the auction, allowing the user with the lowest bid to buy the NFT. This attack can allow two malicious users, the auction winner and the bid shielder, to control the auction and the NFT price [12]. In our solution, all bidders are required to pay for their bids once they place them, and they cannot withdraw from the auction once they have placed their bids. This helps in mitigating the bid shielding attack.

### 2) SMART CONTRACT CODE SECURITY ANALYSIS

Smart contract code, which is written using Solidity, is prone to cyber attacks and can be at risk of many threats and exploitation. Therefore, security tools can be used to check and verify how secure a code is. Our code is written using the Remix IDE, which checks for compilation errors as well as run-time errors. It was not possible to run the full code using a security analysis tool as the code uses built-in functions from the OpenZeppelin library. Using the Oyente tool, we found no bugs in our code. The Oyente tool has certain requirements to run its deep security analysis, such as the solidity version of the code should not exceed 0.4.21 where as the solidity version required by OpenZeppelin is 0.8. Therefore, in order to test our code, we decided to extract some of the functions that can run independently and do not call any functions from other libraries. We tested the code for reputation and rewards. The code was copied into a contract named ''Reputation Smart Contract'' and it was checked to meet the requirements of the Oyente tool, such as a low compiler version. We then ran the code through the Oyente security analysis tool. The tool helps in detecting vulnerabilities such as integer underflow, integer overflow, transaction ordering dependency, and timestamp dependency. It also shows the Ethereum Virtual Machine (EVM) coverage as a percentage. The result generated from the tool is in figure 10. The result shows an acceptable EVM coverage of 74.7% and false for all possible threats.

## *B. CHALLENGES*

The key challenges in our solution are trust and reliability. In order for the reputation system to thrive, all users must be willing to rate each other. They must also do it fairly. Incentives and rewards are an important part of our solution and system design. Both sellers and buyers must act honestly to be rewarded and increase their reputation. The NFTM also plays a major role in building a trusting environment.



```
 0.4.19
INFO:root:contract code.sol:Reputation:
INFO:symExec:   ============ Results ===========
INFO:symExec:     EVM Code Coverage:                     74.7%
INFO:symExec:     Integer Underflow:                     False
INFO:symExec:     Integer Overflow:                      False
INFO:symExec:     Parity Multisig Bug 2:                 False
INFO:symExec:     Callstack Depth Attack Vulnerability:  False
INFO:symExec:     Transaction-Ordering Dependence (TOD): False
INFO:symExec:     Timestamp Dependency:                  False
INFO:symExec:     Re-Entrancy Vulnerability:             False
INFO:symExec:   ====== Analysis Completed ======
root@d740feb3f039:/oyente/oyente# 
```

**FIGURE 10. Security analysis results of our smart contract code from the Oyente tool.**

Users with a better reputation have higher visibility and more privileges in the marketplace. Furthermore, the marketplace must not hide, or hinder the submitted ratings by the users. Multiple gateways can be used to ensure that the ratings are submitted on time. The whole ecosystem could be highly successful if each user fulfilled their role honestly. To enforce trust and also conquer counterfeit NFTs, we changed the metadata information to include the IPFS hash of the NFT posted on the marketplace. The IPFS hash is immutable as it is part of the blockchain logs, which are tamper-proof.

The marketplace must also watch for trends in the sales taking place. Wash trading is a type of collusion attack that is challenging to tackle with proactive measures. The colluding users engage in spurious activities to inflate the market levels, such as demand for an artist or seller or sales volume of a certain collection of NFTs [12]. This is usually detected when a group of users keeps selling to each other the same set of NFTs. However, reactive measures can be used to mitigate such behaviour. Once detected, the user's privileges can be revoked as well as their reputation scores lowered. A proactive measure that can be taken is to ensure that the marketplace always keeps track of all NFT reselling and that the NFT can be tracked when resold to ensure that each owner can only own it once. This helps in avoiding a "chain" loop of illegitimate sales.

Another challenge is scalability. Our solution relies on the Ethereum distributed ledger. Ethereum is in the process of undergoing a revolutionary change to overcome its efficiency issues and increase its speed drastically. Currently, Ethereum 1.0 is a highly secure single-consecutive chain of blocks. But its architectural design has lowered its performance and caused great delays for applications that require more than 30 transactions per second. However, Ethereum's upgrade is extremely promising with its over a hundred thousand transactions per second [28]. Ethereum 2.0 could be a solution to the current network's congestion issues as well as delays. Another huge breakthrough in Ethereum 2.0 is in the way miners validate nodes. Miners currently compete using the proof of work (PoW) to solve a complex mathematical problem. This makes miners compromise energy as well as power. The new Ethereum 2.0 relies on proof of stake (PoS) consensus where miners are randomly assigned blocks. It also incorporates sharding which greatly eliminates delays and increases the network's throughput [29], [30]. It is worth mentioning that proof of stake here in this context refers to the consensus method used in Ethereum 2.0, which is different from staking in reputation systems, where users with a low reputation must deposit a stake or collateral.

### C. GENERALIZATION

Our solution is not limited only to the NFT marketplace. The blockchain-based reputation system can also be included in several other applications. Our solution is adaptable to suit the specifications of different applications that require a reputation system. It is flexible and can be used in other marketplaces and e-commerce. The factors that affect the weight of the ratings can be altered to specifically suit a specific implementation and environment. Our smart contract code can be amended to include more functions or to alter the existing functions based on the needs of the design. In addition to the reputation system, incentivication using rewards and through implementing penalties can be applied in other systems and designs. All new users and users with a low reputation score must provide proof of stake.The Ether withdrawn from the users is returned once a good reputation is established and maintained. This concept incentivizes users to avoid misconduct and can be beneficial in many applications. Furthermore, registration is another aspect that can be used by other solutions. The KYC concept helps in gaining a good insight into the users' background and in maintaining a proper real identity off the chain. It also associates a real identity with all on-chain users and accounts.

## VII. CONCLUSION

In this paper, we proposed a blockchain-based reputation solution for the NFT ecosystem. Our solution leverages the intrinsic security features of blockchain technology, such as data integrity, tamper-proof logs, transparency, accountability, and non-repudiation. Our system design incorporates the decentralized IPFS and stores its hash on the chain to enforce trust in the NFT marketplaces. We developed smart contracts that automate registration, rewards, and incentives. Our design is carefully structured to handle the known NFTM attacks and to eradicate the actions of illegitimate users. The user's reputation is built with time as the users' visibility and privileges in the NFTM increase with their reputation. Our reputation system is decentralized and it depends on weighting factors to compute the aggregated reputation score. All involved factors have been carefully studied to ensure that they combat unfair ratings as well as attacks. The on-chain provenance data includes all the submitted ratings as well as computed reputation scores, which ensures reliability and transparency. Moreover, our solution presents reputation calculation details, tested algorithms, and implementation details with sequence diagrams. We also discussed how our solution is designed to be against known security attacks. We performed security analysis to show that our smart contract code is secure enough against well-known vulnerabilities. We outlined some limitations in the form of challenges that act as future research directions.

## REFERENCES

[1] G. Wang and M. Nixon, "SoK: Tokenization on blockchain," in *Proc. 14th IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–9, Art. no. 11, doi: 10.1145/3492323.3495577.

[2] M. Dowling, "Fertile LAND: Pricing non-fungible tokens," *Finance Res. Lett.*, vol. 44, Jan. 2022, Art. no. 102096.

[3] D. Chalmers, C. Fisch, R. Matthews, W. Quinn, and J. Recker, "Beyond the bubble: Will NFTs and digital proof of ownership empower creative industry entrepreneurs?" *J. Bus. Venturing Insights*, vol. 17, Jun. 2022, Art. no. e00309.

[4] 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, Dec. 2022.

[5] *Swarm*. Accessed: Mar. 23, 2022. [Online]. Available: https://www.larvalabs.com/cryptopunks

[6] 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.

[7] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab, "Blockchain for industry 4.0: A comprehensive review," *IEEE Access*, vol. 8, pp. 79764–79800, 2020.

[8] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.

[9] D. W. Arner, D. A. Zetzsche, R. P. Buckley, and J. N. Barberis, "The identity challenge in finance: From analogue identity to digitized identification to digital KYC utilities," *Eur. Bus. Org. Law Rev.*, vol. 20, no. 1, pp. 55–80, Mar. 2019.

[10] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

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

[12] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, "Understanding security issues in the NFT ecosystem," 2021, *arXiv:2111.08893*.

[13] K. B. Muthe, K. Sharma, and K. E. N. Sri, "A blockchain based decentralized computing and NFT infrastructure for game networks," in *Proc. 2nd Int. Conf. Blockchain Comput. Appl. (BCCA)*, Nov. 2020, pp. 73–77.

[14] R. Dennis and G. Owen, "Rep on the block: A next generation reputation system based on the blockchain," in *Proc. 10th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2015, pp. 131–138.

[15] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic, "IoT public fog nodes reputation system: A decentralized solution using ethereum blockchain," *IEEE Access*, vol. 7, pp. 178082–178093, 2019.

[16] K. N. Khaqqi, J. J. Sikorski, K. Hadinoto, and M. Kraft, "Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application," *Appl. Energy*, vol. 209, pp. 8–19, Jan. 2018.

[17] A. Gutscher, "A trust model for an open, decentralized reputation system," in *Proc. IFIP Int. Conf. Trust Manage.* Cham, Switzerland: Springer, 2007, pp. 285–300.

[18] M. Madine, K. Salah, R. Jayaraman, Y. Al-Hammadi, J. Arshad, and I. Yaqoob, "AppXchain: Application-level interoperability for blockchain networks," *IEEE Access*, vol. 9, pp. 87777–87791, 2021.

[19] M. A. Azad, S. Bag, and F. Hao, "PrivBox: Verifiable decentralized reputation system for online marketplaces," *Future Gener. Comput. Syst.*, vol. 89, pp. 44–57, Dec. 2018.

[20] Z. Zhou, M. Wang, C.-N. Yang, Z. Fu, X. Sun, and Q. M. J. Wu, "Blockchain-based decentralized reputation system in E-commerce environment," *Future Gener. Comput. Syst.*, vol. 124, pp. 155–167, Nov. 2021.

[21] M. Li, L. Zhu, Z. Zhang, C. Lal, M. Conti, and M. Alazab, "Anonymous and verifiable reputation system for E-commerce platforms based on blockchain," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4434–4449, Dec. 2021.

[22] *People are Selling Themselves Their Own NFTs to Drive up Prices, Report Finds*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.nbcnews.com/tech/security/nft-sales-show-evidence-wash-trading-researchers-say-rcna14535?mc_cid=fd65933488&mc_eid=6dab2b9fbb

[23] *Openzeppelin Docs ERC 721*. Accessed: Feb. 22, 2022. [Online]. Available: https://docs.openzeppelin.com/contracts/2.x/api/token/erc721

[24] D. Mouris and N. G. Tsoutsos, "NFTs for 3D models: Sustaining ownership in industry 4.0," *IEEE Consum. Electron. Mag.*, early access, Apr. 1, 2022, doi: 10.1109/MCE.2022.3164221.

[25] Z. Zhou, M. Wang, C.-N. Yang, Z. Fu, X. Sun, and Q. M. J. Wu, "Blockchain-based decentralized reputation system in E-commerce environment," *Future Gener. Comput. Syst.*, vol. 124, pp. 155–167, Nov. 2021.

[26] *Remix*. Accessed: Jul. 27, 2020. [Online]. Available: https://remix.ethereum.org/

[27] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-based distributed trust and reputation management systems: A survey," *IEEE Access*, vol. 8, pp. 21127–21151, 2020.

[28] *What is Ethereum 2.0 and Why Does it Matter*. Accessed: Dec. 7, 2021. [Online]. Available: https://decrypt.co/resources/what-is-ethereum-2-0

[29] *Shard Chains*. Accessed: Dec. 12, 2021. [Online]. Available: https://ethereum.org/en/eth2/shard-chains/

[30] *Ethereum's Latest Progress Toward Proof-of-Stake*. Accessed: Dec. 12, 2021. [Online]. Available: https://www.coindesk.com/tech/2021/10/13/ethereums-latest-progress-toward-proof-of-stake/