2022 International Conference on Culture-Oriented Science and Technology (CoST) | 978-1-6654-6248-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/CoST57098.2022.00041

A New NFT Model to Enhance Copyright Traceability of the Off-chain Data

Yulong Chen, Ziwei Wang, Xiangyu Liu, Xuetao Wei
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
12031272@mail.sustech.edu.cn, 11510838@mail.sustech.edu.cn, liuxy33@sustech.edu.cn, weixt@sustech.edu.cn

***Abstract*—Non-Fungible Tokens (NFTs) are digital assets that represent real-world objects like art, music and videos. However, NFTs according to current standards have no provisions for the copyright traceability of the off-chain data, which greatly hinders the sustainability of the NFT community. In this paper, we propose a new NFT model, which is a synergy of a new economic mechanism backed by game theory and two supplementary algorithms to handle the off-chain data. The economic mechanism is first proposed to motivate participants to maintain the off-chain raw data. Then, the model includes two supplementary algorithms, the version algorithm and validation algorithm, to verify the NFT's ownership and copyright. We implement our model in Solidity on Ethereum and conduct experiments based on the real-world dataset from the largest NFT marketplace OpenSea. Our evaluation demonstrates that our model is a promising attempt towards the copyright traceability of the off-chain data for NFTs.**

***Keywords*—blockchain, NFT, model, copyright**

I. Introduction

Since Ethereum [1] appeared, smart contracts have been applied to various scenarios. For example, in Ethereum, you can issue your own tokens following the ERC20 standard [2]. Tokens following the ERC20 standard are fungible tokens (FTs). FTs mean that the same quantity of tokens owned by different owners has exactly the same value with no difference. FTs are interchangeable and divisible, which is similar to our real fiat currency [3]. In contrast, some assets are unique and indivisible, which are called as non-fungible tokens (NFTs). For example, an application called CryptoKitties [4] on Ethereum, which is based on the ERC721 standard [5].

Due to the high cost of storage on the blockchain, only the necessary description information of NFTs (e.g., the owner and the ID of an NFT) is stored on the blockchain. The items that NFTs represent, e.g., images, animated GIFs and videos, are generally stored off the blockchain. As NFTs currently do not have any incentive mechanism to guide participants to maintain these data, they are at risk of loss [6]. Most of the off-chain data is stored in the public storage (e.g., IPFS), which means anyone can use these data without any permit, and it is difficult for the owner to trace who has carried out the relevant illegal operations.

Lack of maintenance of off-chain data makes them easily leaked and abused. Given that the value of each NFT lies in its uniqueness and scarcity, addressing these problems mentioned above is extremely urgent for the sustainability of the NFT community. However, no work discussed the copyright traceability of the off-chain data for NFTs up to now. Some work [7], [8] discussed certain enhancements to the NFT infrastructure, but these enhancements are not about off-chain data. In industry, various NFT standards [5], [9]–[13] were proposed by different blockchain communities, but they mainly aimed at unifying the most basic functions of NFTs (e.g., the transfer function) and had no provisions for the copyright traceability of the off-chain data.

In response to these problems, we propose a new NFT model to enhance the copyright traceability of the off-chain data. In our model, participants' behavior is guided by economic mechanisms. These mechanisms backed by game theory [14] can motivate involved participants to protect the off-chain data and make each owner has a distinct and verifiable version of off-chain data. We implement our NFT model with Solidity on the Ethereum. We evaluate our NFT model with the real-world dataset from the largest NFT market OpenSea. Our evaluation shows the proposed new model is feasible. Overall, our work is a promising attempt towards the copyright traceability of the off-chain data for NFTs, which could significantly improve the sustainability of the NFT community.

II. Model Overview

A. Terms

We introduce some concepts and terms used in this paper:

Raw data. The original off-chain data created by NFT creators without any owner's information embedded in are called raw data.

Specific version data. Specific version data is generated for each NFT owner. Specific version data is embedded with the owner's unique information from the raw data by a certain algorithm, which makes them distinguishable.

Version algorithm. The algorithm that generates specific version data from raw data for a specific NFT owner is called the version algorithm. The version algorithm needs to meet certain requirements (see Section IV-A).

Validation algorithm. It is used to verify whether the specific version data is derived from the real raw data. The validation algorithm needs to meet certain requirements (See Section IV-B).

978-1-6654-6248-8/22/$31.00 ©2022 IEEE
DOI 10.1109/CoST57098.2022.00041

Validation code. The result of using the validation algorithm to process the off-chain data is called validation code. It's the fingerprint of the off-chain data and representation of the off-chain data.

NFT manager. In order to realize that each NFT owner has a specific version of the off-chain data, the raw data created by the creator should be protected. So it is necessary to select a person or organization to take this responsibility. We believe that this responsibility will naturally fall on the NFT creator. Because he is the one who has created and authored the raw data. The role that maintains the raw data and generates specific version data for the owners is called the NFT manager.

NFT owner The person who has bought the NFT is called the NFT owner (or owner for short).

B. Overview

Our NFT model includes four modules: (a) management module, (b) incentive module, (c) penalty module, and (d) off-chain data generation and validation module. The entire architecture of our model is shown in Figure 1. Under normal circumstances, the process of an NFT from creation to finishing trade is as follows:

(1) The NFT manager is registered in the manager contract. Then he deploys an NFT contract and deposits a certain amount of tokens. The NFT contract deployed by the NFT manager is also registered in the manager contract.

(2) Then NFT manager issues a new NFT with the validation code of the raw data stored on the contract. Then he puts it on sale with a price.

(3) Before deciding to buy an NFT, buyers may check the NFT's transaction records and the reputation of the NFT's manager (i.e., the NFT creator in this paper), doing these will help them make the right decision.

(4) After the purchase transaction has been submitted by a buyer, the NFT manager should generate the specific version data for the new owner and update the information (e.g., validation codes) of the specific version data on the contract. The NFT manager is willing to do all these because of both the incentive module and the penalty module.

(5) After getting the specific version data, the new owner can verify whether the specific version data is derived from the real raw data according to the validation codes of the specific version data and that of the raw data in the NFT contract. The owner can also verify the specific version data is relevant to him by extracting the information from them using the version algorithm. After verifying all these things, the new owner can confirm the data in the NFT contract to get back some of the tokens he paid when bought the NFT.

III. Economic Mechanisms

In our model, the specific version data generated for the specific owner is only sent to the corresponding owner and maintained by himself. The NFT manager only needs to maintain the raw data, which is motivated by our economic mechanisms. s

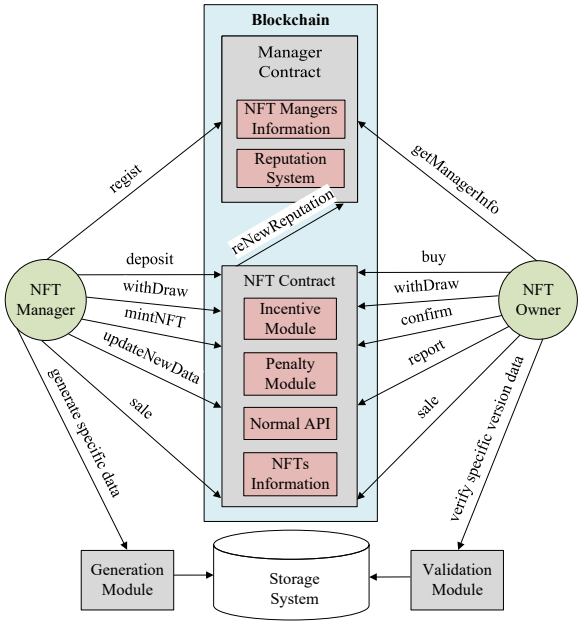

Fig. 1. The Architecture of the New NFT Model

The details of the economic mechanisms (including the incentive and penalty mechanisms) are shown in the Figure 2 and described as follows:

(1) When the NFT manager creates an NFT, his deposit must be more than the specified value called $minManagerDeposit$ in the NFT contract, or he fails in creating an NFT.

(2) When a buyer buys an NFT, in addition to paying for the NFT itself, he also needs to pay for the specific version data generated by the NFT manager, which value is denoted as $newDataFee$. Meanwhile, the NFT buyer needs to submit a confirmation deposit called $confirmDeposit$ to ensure that the buyer confirms whether he has received the specific version data.

(3) When the NFT is bought by someone, the NFT manager should generate specific version data for the buyer, and he can get a certain reward denoted as $newDataFee$ which is specified in the contract. If the NFT Manager does not finish it within a time (measured by the number of blocks), the buyer can report and a part of the NFT manager's deposit denoted as $compensation$ is forced to transfer to the buyer's account as the compensation.

(4) If the NFT manager has generated the specific version data for the new owner, the buyer should confirm it and withdraw the deposit $confirmDeposit$. If the buyer does not confirm but report, the report is considered malicious and the buyer's deposit is forced to transfer to the NFT manager.

(5) In order to track the behavior of the NFT managers (i.e., the NFT creators), a manager contract is designed. All NFT managers should register in this contract. When NFT managers act according to the rules, the NFT contract increases their reputation, and when NFT managers do not act according to the rules, their reputation is decreased.

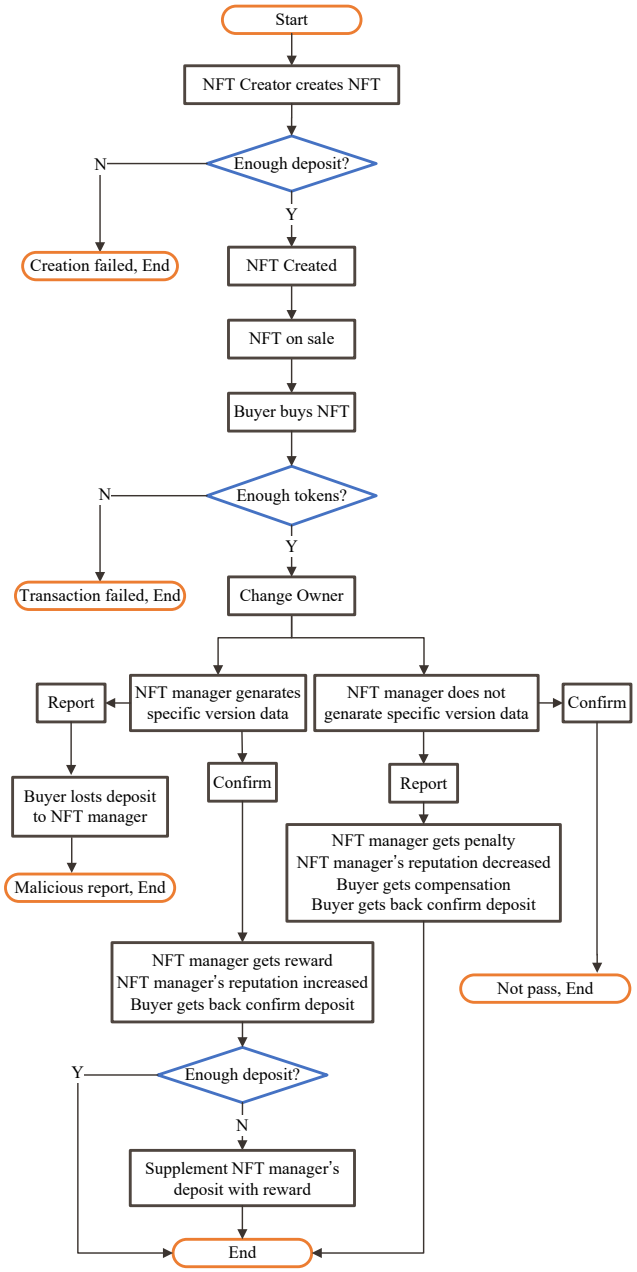

Fig. 2. Mechanism Diagram

Note that, these mechanisms mentioned above are fixed in the smart contract, so they can be enforced eventually.

A. The Analysis of Economic Mechanisms

Throughout an NFT's whole life cycle, we divide its period into two parts, which are as follows:

(1) When the NFT is being traded, that is, a buyer has sent a transaction to buy the NFT. We call this situation scenario 1.

(2) When the NFT is not being traded. In this scenario, no buyer is buying the NFT, and there are only potential buyers. We call this situation scenario 2.

We abstract the situations of these two scenarios into games and use the game theory to show that under the mechanisms given above (refer to the Section III), rational NFT managers would play by the rules as intended.

1) Analysis of the scenario 1: Scenario 1 is abstracted into a general strategic game and the details are as follows:

Players. In this scenario, there are two kinds of players: the NFT manager and the buyer.

Actions. The buyer has 2 strategic choices: confirm or report the specific version data. The NFT manager has 2 strategic choices: generate or not generate specific version data for the buyer.

Players' Revenues. The NFT manager has 2 strategies to choose from, and the buyer has 2 strategies to choose from, forming a total of 4 situations.

The revenues of the buyer are analyzed as follows:

(1) When the NFT manager chooses the strategy of generating the specific version data, then: (a) if the buyer chooses to confirm the specific version data. Given that the buyer pays for the NFT and he gets it, he neither gains nor loses, the revenue is 0; (b) if the buyer chooses to report, since the NFT manager has generated the specific version data, the report is considered malicious in the smart contract, and the buyer's confirmation deposit is confiscated. The revenue is negative and recorded as $-a1$ ($a1 > 0$).

(2) When the NFT manager chooses the strategy of not generating the specific version data, then: (a) if the buyer chooses to confirm the specific version data. The confirm action is not passed. Since the buyer only gets the incomplete NFT, his revenue is negative and recorded as $-a2$ ($a2 > 0$); (b) if the buyer chooses to report, since the NFT manager does not generate specific version data, the report is passed in the smart contract and the buyer gets the compensation which is recorded as $a3$ ($a3 > 0$). However, the NFT that the buyer gets is incomplete, so he still loses $a2$. Thus, the total revenue is $(a3 - a2)$.

The revenues of the NFT manager are analyzed as follows:

(1) When the buyer chooses the strategy of confirming his specific version data, then: (a) if the NFT manager generates the specific version data, then he obtains a reward. The revenue is positive and recorded as $b1$ ($b1 > 0$); (b) if the NFT manager does not generate the specific version data, the buyer's confirm action is not passed. The NFT manager can't get the reward, and his revenue is recorded as 0.

(2) When the buyer chooses the strategy of reporting, then: (a) if the NFT manager generates the specific version data, The buyer's report fails, besides the owner's deposit is transferred to the NFT manager for the owner's malicious report. It means that the NFT manager gets some tokens, and the NFT manager's revenue is positive and recorded as $b2$ ($b2 > 0$); (b) if the NFT manager does not generate the specific version data, the buyer's report is passed and the NFT manager is penalized, his revenue is recorded as $-b3$ ($b3 > 0$). Besides, there is an impact on the NFT manager's reputation, which

TABLE I. The Matrix Form: Game of Scenario 1

Buyer \ NFT manager	Generate	Not generate
Confirm	(0, b1)	(-a2, 0)
Report	(-a1, b2)	(a3-a2, -b3-b4)

* $a1 > 0, a2 > 0, a3 > 0, b1 > 0, b2 > 0, b3 > 0, b4 > 0$.

TABLE II. The Matrix Form: Game of Scenario 2

Potential Buyer \ NFT Manager	Protect	Expose
Buy	(0, d1)	(-c1, d1-d2)
Not buy	(0, 0)	(0, -d2)

* $c1 > 0, d1 > 0, d2 > 0$.

brings some losses recorded as $-b4$ ($b4 > 0$), so the total revenue of the NFT manager is $(-b3 - b4)$.

We put the game of scenario 1 into the matrix form as shown in Table I. We find (0, $b1$) is the only pure strategy Nash equilibrium point of the game, from which the following conclusion can be drawn: after the buyer has made a purchase, in order to maximize their interests, the NFT manager should generate the specific version data for the buyer and the buyer should confirm it.

2) Analysis of the scenario 2: In the same way, we put the game of scenario 2 into the matrix form as shown in Table II. $d1$ is the potential profit of the NFT manager if his NFT is sold. Where, $-c1$ is the loss of the buyer if he buys an NFT that the NFT manager does not protect the raw data. $-d2$ is the negative impact on the NFT manager if he does not protect the raw data.

Now we can see that, the game does not have a unique Nash equilibrium point, but the NFT manager has a dominant strategy: No matter whether the buyer buys or not, the NFT manager should adopt the same strategy (i.e., protect the raw data) to maximize his interests.

3) Summary: Combining the results of the game theory analysis of both scenario 1 and scenario 2, we can observe that throughout the whole life cycle of the NFT, all parties will behave the way we expect.

IV. Supplementary Algorithms

In this section, we specify the requirements of the supplementary algorithms in our model, The requirements can guide us how to choose the algorithms. Due to the space limitation and for the sake of illustration, we take the example of the off-chain image data and we show how to choose algorithms that satisfy the requirements of our model.

A. The Version Algorithm

Our purpose of using the version algorithm can be summarized into two parts: (a) let each owner of an NFT has specific and distinguishable off-chain data and (b) Protect raw data from disclosure. Based on these, we can summarize the features that the version algorithm requires are as follows:

(1) It can embed the unique information of the owner into the specific version data;

(2) It is impossible (very difficult) to recover the raw data from the specific version data;

(3) As with the raw data, the specific version data will not affect the normal use and appreciation.

Intuitively, we can use the watermarking algorithm [15] as the version algorithm in our model to process images. We embed the owner's account information into the image as the watermark, so as to meet the first requirement above. We choose a watermarking algorithm that is difficult to crack, so that the raw data is not exposed, which meets the second requirement. Finally, changes in the watermarked image are not perceived. It basically does not affect the normal use, which meets the third requirement.

B. The Validation Algorithm

Our purpose of using the validation algorithm is to verify the specific version data embedded with the unique information of different owners is derived from the raw data. The validation algorithm requires the following features:

(1) The owner can verify the specific version data without having to get the raw data;

(2) The validation code(refer to II-A) of the algorithm is small enough for storing in blockchain;

(3) The validation results are not fuzzy and have good discrimination.

Here, we continue to take the image as an example. We can use the algorithm that can calculate the similarity of two images as our validation algorithm. That is, we calculate the similarity of the original image (i.e., the raw data) and the specific version image. If they are highly similar, the specific version image is considered to be generated from the original image. Specifically, we can use the perceptual hash [16] as the validation algorithm for images. The perceptual image hash algorithm calculates the fingerprint of the image as the validation code, which is very small (e.g., 64 bits). The NFT manager puts the fingerprint of the original image (i.e., raw data) on the NFT contract when he creates an NFT. After the NFT is transferred to a new owner and the new owner gets his specific version data. Then, he can calculate the fingerprint and compare it with the fingerprint of the raw data recorded on the contract to verify if his specific version data is derived from the raw data.

C. Summary

Overall, when we select the version algorithm or validation algorithm, the most important thing we should concern is that it must meet the corresponding requirements specified above.

V. Implementation and Evaluation

We implement a demo of our model with Solidity and here we explain our implementation in detail.

A. NFT states

Our NFT has four states: $offSale$, $onSale$, $saledStart$ and $saledEnd$. An NFT can only be in one state at a time throughout its life cycle. An NFT's state is $offSale$ when

created and the owner can change it to $onSale$ by putting it in the on-sale list. When the NFT's state is $onSale$, a buyer can buy it and change the NFT's state to $saledStart$. When the NFT's state is $saledStart$, that means the NFT's owner has been changed to the buyer, but the buyer has not got his specific version data. The NFT manager can update the information of the specific version data and change the NFT's state to $saledEnd$, and then the buyer confirms the specific version data and changes the NFT's state back to $offSale$. The transitions between states of an NFT are shown in Figure 3.

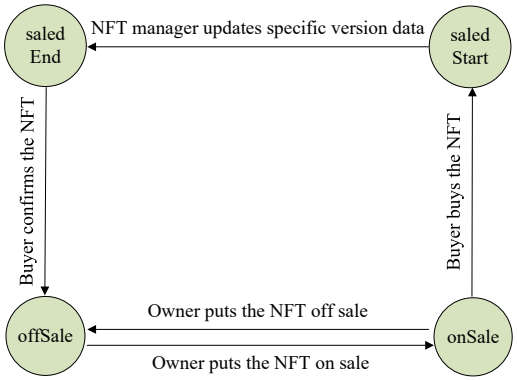

Fig. 3. Transitions of NFT States

B. Important Variables

As our model has more features, so there are some extra variables. Before introducing them, we first introduce the struct of NFT (an instance of which is an NFT), which contains some important fields as follows:

vCode: The validation code of the raw data.

svCode: The validation code of owner's specific version data.

transferBNum: The time (in the form of the block number) when the NFT was transferred to the owner.

svCodeBNum: The time (in the form of the block number) when the NFT manager updates the information of specific version data.

Some other fields in the NFT struct are easy to understand and are not listed here due to the limited space. Now, we introduce some important extra variables in our model as follows:

NFTs: This is the mapping from the token IDs to the NFT instances. This mapping stores all the NFTs.

onSaleList: This is the mapping from the token IDs to the NFT instances that the states are $onSale$.

needNewDataList: This is the mapping from the token IDs to the NFT instances that the states are $saled$.

withdrawPool: This is the mapping from the addresses to balances that store the number of tokens accounts can withdraw.

C. Important Functions

Important functions that are unique to our NFT model are detailed here as follows.

Create an NFT: In our model, the creation of an NFT needs to meet certain condition that the deposit of the NFT Manager needs to be greater than a specified value (denoted as $minManagerDeposit$), which is different from the commonly used contract standard(e.g., ERC721).

After the NFT is created, the NFT's initial state is $offSale$. The owner can put it on sale, which makes the NFT's state change to $onSale$. The Owner also should specify the price in this action.

Buy an NFT After the NFT is put on sale, a buyer can buy the NFT. As we mentioned above, in addition to pay for the NFT itself, the buyer should pay for the specific version data and submit a deposit for the confirm action. An event (i.e., $NFTTransferEvent$) is emitted and the block number (denoted as $transferBNum$) is also recorded in this function.

Update the Information of Specific Version Data: The NFT manager should subscribe to the event $NFTTransferEvent$. When he captures the event, he should generate the specific new data for the new owner and use the validation algorithm to calculate the validation code and update the $svCode$ and uri of the NFT within a specified period of time (denoted as $maxNewDataBNum$). After the update, the NFT's state is transferred to $saledEnd$ and event $updateNewDataEvent$ is emitted.

Confirm the Specific Version Data: When the buyer captures the event $updateNewDataEvent$, he can get his specific data through the new URI. Then, he can use the validation algorithm to calculate the validation code and verify if the specific data is derived from the raw data. If something goes wrong, the owner should contact the NFT manager (the contact information is posted in the contract) to fix the problem. The NFT manager is willing to cooperate with the new owner because the NFT manager can not get his reward before the new owner confirms the specific version data. When everything is done, the owner should invoke the confirm function, through which he can get back his deposit $confirmDeposit$.

It is worth pointing out that when the deposit of the NFT manager is insufficient, his reward is transferred to his deposit, which is realized by the withdraw function (i.e., limit withdrawals when the deposit is too low). So in the confirm function, the reward is directly transferred to the NFT manager.

Report the Specific Version Data: If the NFT manager does not update the specific version data on time (i.e., after $maxNewDataBNum$ block numbers from $transferBNum$), the owner can report the fact by calling the report function on his behalf. The report function does not change the NFT's state to $offSale$, but leaves it to remain $saledStart$. The NFT manager can still update the information of specific version data, but he couldn't get back the fine he gave to the owner. If the NFT Manager has generated specific version data as required but the owner still reports, that is considered a malicious report and his deposit is mandatory to be transferred to the NFT Manager.

TABLE III. Probability of different hash distances

Distance	0.04	0.05	0.1	0.2	0.65
Probability	91.86%	93.13%	96.05%	97.28%	100%

D. Experiment

We have analyzed that the economic mechanism of our model is feasible in theory (refer to Section III-A). Now, the most important thing we should evaluate is whether the algorithms currently available for handling the off-chain data are feasible, which is also important to enable our whole model to work. To test that, We randomly download more than 1 million images from OpenSea [17], the largest NFT platform. Then we select the most commonly used image watermarking algorithm [15] based on the Discrete Cosine Transform(DCT) [18] as the version algorithm and the perceptive hash algorithm [16] as our validation algorithm.

1) Version Algorithm: For the downloaded 1,066,696 images (raw data), we randomly generate a 40-bit Ethereum account information in hexadecimal, and then use the version algorithm to embed the account information into these images to generate specific version data. We then try to extract account information from these version-specific data and compare them with the account information at the time of embedding. The results show that there are 998,916 images that can successfully extract correct account information, with an accuracy rate of 93.65%.

2) validation Algorithm: In addition, for the 998916 images that have generated corresponding specific version data correctly above, we use the perceptive hash (256bit) algorithm as our validation algorithm to verify whether 998916 specific version data are from the corresponding raw data. We use the normalized hamming distance defined in [16] to represent the similarity of the two images. Figure 4 shows the normalized hamming distance distribution and Table III shows some specific points' value of the distribution function. We can see that, when the threshold of judging the normalized hamming distance is set to be very low, there's still a high probability that algorithms work. For example, when we set the judging threshold as 0.1, the algorithm can be applied to 96.05% of the images.

3) Summary: Overall, we can choose the appropriate algorithms that well fit into our model's requirements. Though we use the image as an example to showcase the model usage, the same spirit can be applied to other off-chain data of NFTs, e.g., giFs or videos. Therefore, our analysis shows that our model is feasible.

VI. Conclusion

In this paper, we have proposed a new NFT model to enhance the copyright traceability of the off-chain data. Our analysis based on game theory and evaluation based on the real-world datasets have shown that our proposed NFT model is a promising attempt for the copyright traceability of NFTs' off-chain data.

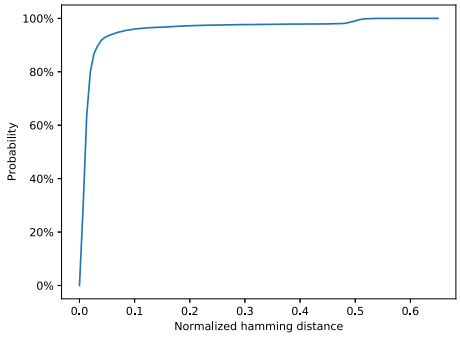

Fig. 4. The Distribution Function of the Hamming Distance Distribution

Acknowledgment

This work was supported in part by National Key R&D Program of China under Grant 2021YFF0900300. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[2] V. B. Fabian Vogelsteller, "Erc: Token standard 20." Website, 2022. Accessible: https://github.com/ethereum/EIPs/issues/20.

[3] "United states dollar in wikipedia." Website, 2022. Accessible: https://en.wikipedia.org/wiki/United$_{S}tates_{d}ollar/$.

[4] "Cryptokitties web." Website, 2022. Accessible: https://www.cryptokitties.co/.

[5] J. E. N. S. William Entriken, Dieter Shirley, "Erc: Token standard 721." Website, 2022. Accessible: https://github.com/ethereum/EIPs/issues/721.

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

[7] H. Takahashi and U. Lakhani, "Voting blockchain for high security nft," in *IEEE GCCE*, 2021.

[8] D. Chirtoaca, J. Ellul, and G. Azzopardi, "A framework for creating deployable smart contracts for non-fungible tokens on the ethereum blockchain," in *IEEE DAPPS*, 2020.

[9] W. Radomski, "Erc: Multi token standard 1155." Website, 2022. Accessible: https://github.com/ethereum/EIPs/issues/1155.

[10] "dgoods standard." Website, 2022. Accessible: https://docs.eosstudio.io/contracts/dgoods/standard.html.

[11] "Algorand standard assets (asas)." Website, 2022. Accessible: https://developer.algorand.org/docs/get-details/asa/.

[12] "Fa1.2 asset smart contracts." Website, 2022. Accessible: https://tezos.gitlab.io/user/fa12.html/.

[13] "Flowdocs. non-fungible tokens." Website, 2022. Accessible: https://docs.onflow.org/cadence/tutorial/04-non-fungible-tokens/.

[14] M. J. Osborne *et al.*, *An introduction to game theory*, vol. 3. Oxford university press New York, 2004.

[15] C. Kumar, A. K. Singh, and P. Kumar, "A recent survey on image watermarking techniques and its application in e-governance," *Multimedia Tools and Applications*, vol. 77, no. 3, pp. 3597–3622, 2018.

[16] C. Zauner, "Implementation and benchmarking of perceptual image hash functions," *Accessible: http://www. phash. org/docs/pubs/thesis_zauner. pdf*, 2010.

[17] "Opensea platform." Website, 2022. Accessible: https://opensea.io/.

[18] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.