SBT Report - Article Copyright Recording and Trading System Based on Blockchain

Xinran Lu (flr970), Jingyu Huang (njf735)
Yufei Yuan (ctq566), Xinyi Huang (jbm927)

1 Introduction

In the nowadays world, as the Internet is getting better and better, people are more interested in uploading their works, such as music, articles, and paintings, to the network. Since everybody can download the pieces of others, and post them on his or her own account, we will not have an efficient way to find the real author. Therefore, it means that netizens' artworks will be stolen more easily by other users ([Gui+20]). And people need a useful method to guarantee their own copyright.

In the traditional copyright registration process, the author needs to register his own copyrights with authoritative third parties. This may bring two main problems. First of all, as we all know, authors may spend a large amount of money and time doing the registration. Furthermore, there will be a period between the time that writers' pieces are finished and the time that they are authorized.

To deal with the above challenges, the blockchain technology is applied in the copyright field ([ZO18]). If authors store their work on the blockchain, the content can be updated in time and cannot be tempered with. Apart from this, the timestamp in the blockchain can represent the precise time that the works are finished, and the timestamp cannot be changed. Thus, it is possible for writers to find people who have downloaded their works and use the timestamp as the evidence for the prosecution of plagiarists. ([Eur17]

To ensure the authority of the copyright system, it is essential to have a large number of authors and piece-storage. However, in the existing blockchain for copyright, there is no useful way to attract a sufficient number of users. Therefore, we want to create a community to draw more authors to register their article copyrights.

In general, people can buy and sell copyrights in our community using the ETH, which is a very reliable and convenient way to make a transaction. In addition, both buyers and sellers can be awarded with the ETH or a coin which is unique to our system.

2 Our exploration of blockchain applications for copyright

Two kinds of tokens are introduced in this copyright recording and tracing system:

1. ACoin: Non-Fungible Token(NFT). It is used to present copyrights. When an author uploads a copyright in this system, this author is the copyright owner.

2. CCoin: Fungible Token(FT). This kind of token can only be used in our system as commission fee waiver points.

At first, CCoin can be obtained in two ways.

1. Likes: When users like a work, they can get a certain amount of coins by giving it likes.

2. Transactions: When a buyer buys an NFT, the seller can get a certain amount of CCoin as a reward, and this process is called mining. Users conduct copyright transactions by transferring NFTs. This approach will encourage trading and accelerate the circulation of CCoin.

In general, CCoin must comply with the following rules in this system:

1. Trading ACoin generates CCoin

2. CCoin can be used to reward copyrights

3. CCoin can be used to buy copyrights. The value of copyright is determined by the market.

In our design, NFT could be purchased not only by CCoin but also by other virtual currencies, such as ETH. However, the introduction of ETH would cause CCoin to lose its significance for purchase, so after the first feedback, we decided to reset CCoin and modify the strategy.

In our advanced system, We keep the definition of ACoin and CCoin unchanged. And our modified strategies are shown as following:

1. Buyers need to pay additional commission fee to the system when they purchase other's NFT.

2. Users are not allowed to get CCoins by giving likes to NFTs. Since likes cost nothing and users can easily get CCoins, which can lead to some people maliciously swiping CCoins.

3. Buyers can only use ETH to buy NFT.

4. A new strategy to reward authors: the top 3 copyright seller will be given ETH according to the cumulative number of copyrights sold by them.

And the new rules for CCoin are listed as follows:

1. Trading ACoin generates CCoin.

2. The amount of Ccoin that can be minted from the system per day is limited. Once the number is exceeded, continuing to purchase ACoin will not result in new CCoin.

3. A number of ACoins can waive the commission fee.

By designing the strategy of rewarding authors and using CCoins to waive commission fees, the system can attract more users.

3 Detailed Scheme Design

3.1 Functional Design

3.1.1 Usecase Diagram

Figure 1: usecase diagram.png

The above diagram (Figure 1) shows the actions that can be performed by users with different identities (author and buyer) in our system.

From the timing diagram below (Figure 2), we can see the information transfer process between the user, the front-end, and the blockchain when the user performs different actions.

3.1.2 Timing Diagram

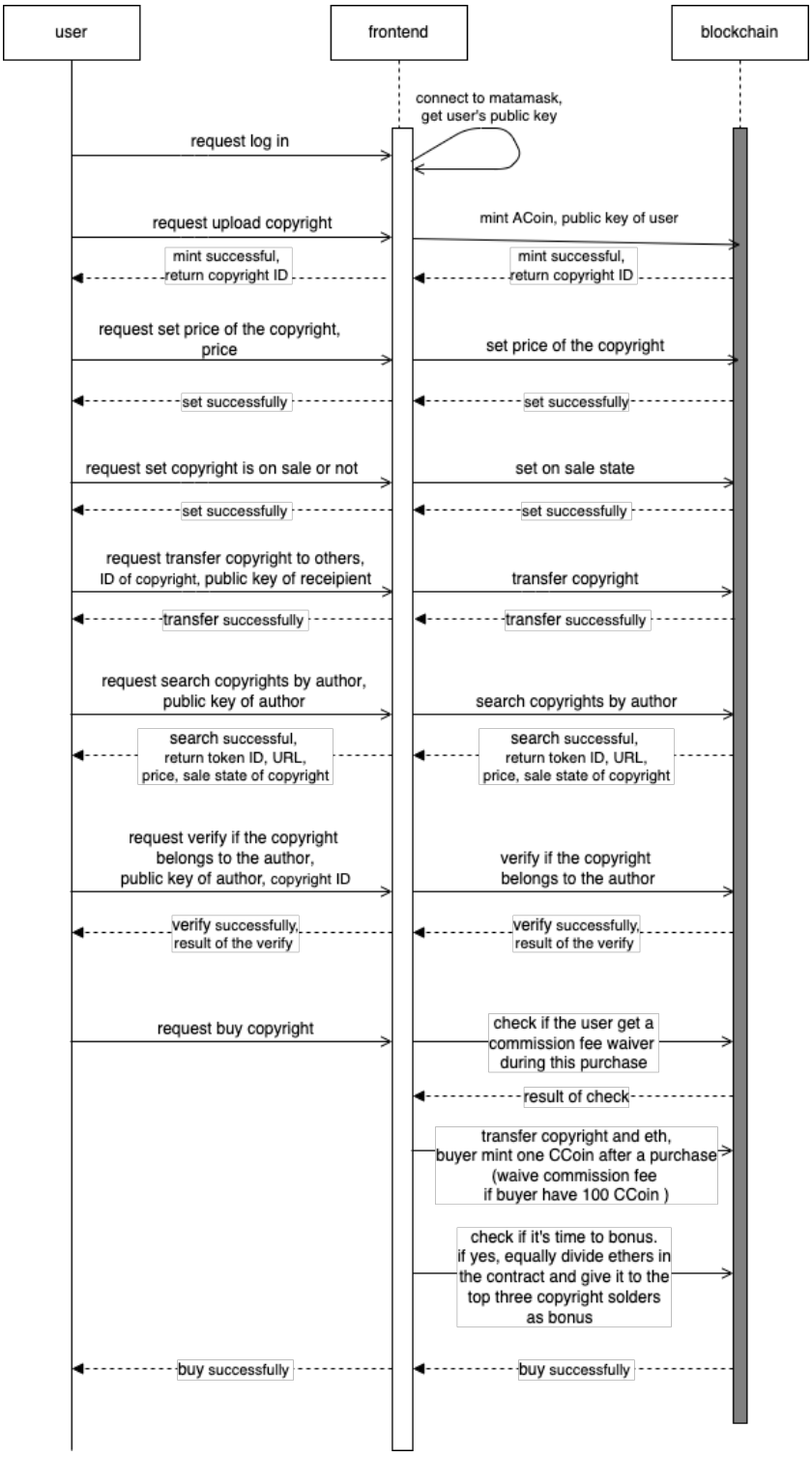


Figure 2: timing diagram.png

3.2 Contract Design

Following is the design of the functions in our ACoin contract and CCoin contract.

3.2.1 ACoin-NFT

Table 1: ACoin Function Table

Function Name	Explanation	Comments
constructor	Construct ACoin	Construct CCoin as well. This is the only way CCoin can be constructed to secure the CCoin function.
mintNFTAnyone (string memory tokenURI)	User mint a ACoin with copyright URL	
mintNFTWithMD5 (string memory tokenURI, string memory MD5)	User mint a ACoin with copyright URL and artical content	Articles content are encrypted with md5 to store.
transfer (address recipient, uint256 tokenId)	Owner transfer the ACoin to other address, the owner need to pay for the commission.	
purchase (uint256 tokenId)	User purchase copyright by copyright ID	Please see specific explanation under this table.
_refund (address recipient, uint256 price, uint256 commission, uint256 paid)	Refund the extra ETH the buyer paid and the commission fee if it's need to be waive.	private function called by function purchase
_giveBonus (address user)	give ethers in contract to the top three copyright solders as bonus	private function called by function purchase. Use a three-digit array to store the addresses of the users of bonus, and refresh them in the function purchase.
setForSale (uint256 tokenId, bool forSale)	Set if it's on sale to true or false by copyright ID and sale state.	
setPrice (uint256 tokenId, uint256 price)	Set the price of the NFT in ETH by copyright ID and price.	

In the function purchase (uint256 tokenId):

buyer need to pay commission fee during purchase. The buyer mints one CCoin after per purchase. And if the buyer has 100 CCoin, he will get a commission waived automatically and lose 100 CCoin by refunding the commission after purchase. Then it will check if it's time to bonus. if yes, equally divide ethers in the contract and give it to the top three copyright solders as bonus.

3.2.2 CCoin-FT

Table 2: CCoin Function Table

Function Name	Explanation	Comments
mintFT (address minterAdd)	minterAdd mint a CCoin	only ACoin can call this function
totalBalance (address account)	get balance of CCoin by public key	
reduceBalance (address account, uint256 amount)	reduce some amount of CCoin by public key	transfer the reduced CCoin to contract address, only ACoin can call this function
mintTransferCCoin (address to, uint256 amount)	transfer some amount CCoin from user to address "to"	only ACoin can call this function

4 Implementation

4.1 Smart Contracts

4.1.1 ACoin

Each ACoin represents an article's copyright. Anyone can mint an NFT, set its price, and decide whether it could be sold. The default value for price and isOnSale is 0 and false, respectively. Once the price is set and the copyright is allowed to sell, anyone could purchase it without the owner's permission by sending enough ethers to the contract's payable function purchase().

Once a purchase happens, the buyer will receive a newly minted CCoin as bonus, and the sales number of the seller increment 1. Within a fixed time period like 1 week, creators who sell the most copyrights will get a bonus, which comes from the commission the contract collect before.

The owner of NFT can transfer it to anyone else by calling the 'transfer()' function, and both the recipient and the owner do not need to pay for the copyright. In case someone pays more than the set price of ethers to the contract, we also provide the refund function, which will automatically calculate and send back extra money.

4.1.2 CCoin

To avoid abusive CCoin mining for malicious bypassing of commission, we added a modifier called onlyACoin, which controls CCoin by checking if the message sender is the ACoin Contract, while allowing external calls.

4.2 Deployment

4.2.1 Deployment environments

Mainnet is where people actually make transactions. Testnet, as we understand it, is an online ethernet chain where developers test their smart contracts before they are deployed to Mainnet. Any cryptocurrencies in the testnets are worthless, and we could actually apply for some free ethers on some so-called faucet websites.

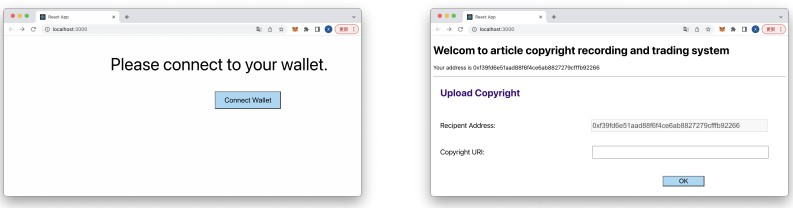
Hardhat Network is a locally-running ethernet chain. It creates 20 test accounts where each account holds 10000 ETH. Therefore, it would be much easier and more convenient for us to develop and test smart contracts. After comparing the different deployment environments, we finally deploy our smart contracts on the Hardhat Network.

4.2.2 Deployment schemes

As CCoin has been downgraded from a full-featured cryptocurrency to a point-like cryptocurrency, a key issue is how to achieve mutual trust between ACoin and CCoin. If we deploy the two contracts separately, then the two contracts would not know where the other is, which requires a complicated authentication mechanism.

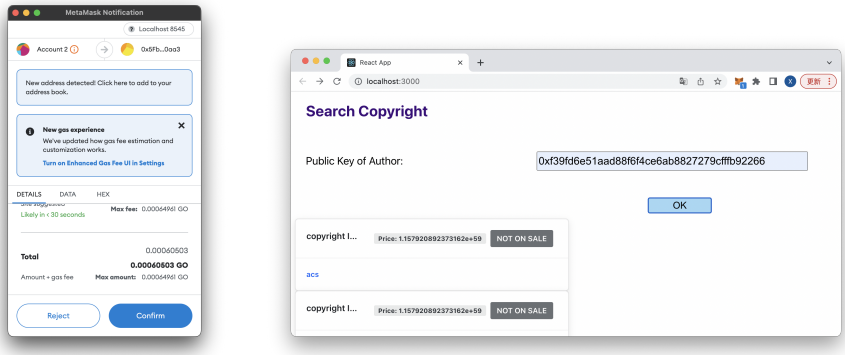
Our solution is to let ACoin Smart Contract create CCoin Smart Contract in its constructor function and pass its deployment address to CCoin contract instance. In this way, the two contracts are still deployed to two different addresses, but ACoin calls the functions of CCoin through the instance it creates and stores, while CCoin accepts the call of CCoin since it stores the address of the ACoin contract, which makes it possible to check if the msg.sender is the ACoin contract that creates it.

4.3 Frontend



(a) log in page (b) mint NFT function

Figure 3: Frontend pages



(a) MetaMask confirm page

(b) search page

Figure 4: Frontend pages

We create a simple frontend as well, to show our use-cases.

Connect Wallet: the homepage of the system. Before execute other functions, it is required to connect wallet first.

Upload Copyright: enter the copyright uri to mint a copyright for the wallet owner.

Verify Copyright: enter the public key of author and copyright ID to verify whether this copyright belong to this author or not.

Set price: enter the copyright ID and the desired price to set the NFT price. This function can only be used to set the wallet owner's NFT price.

Transfer Copyright: enter the copyright ID and recipient address (public key) to transfer NFT. After pressing the button, it will activate the wallet page to pay the fee.

Search Copyright: enter the public key of the author, then it returns this author's copyright information (including the copyright ID and article's url). If author searches own copyrights, the author can change copyright's sale state in the search result page.

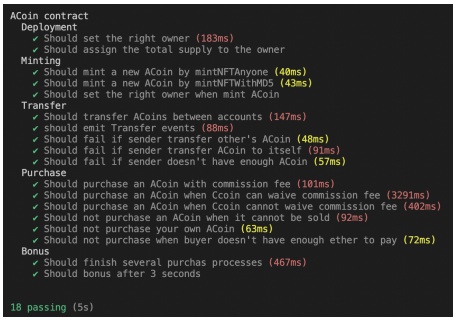
4.4 Tests

To debug our smart contracts and also verify the functionality we envisioned, we design two sets of unit tests using chai test framework. One for ACoin and one for CCoin.

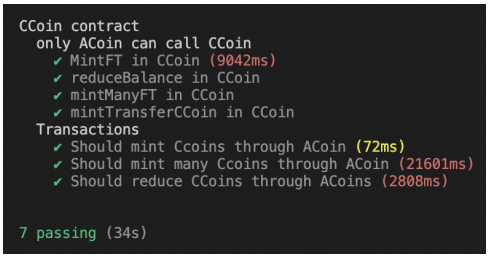
For ACoin, we mainly test the purchase and bonus functions. The purchase function is mainly tested to see whether the commission fee can be waived by the CCoins. For the bonus function, we first get five signers. Then let them purchase ACoins with each other. Finally we check the balance of those signers who purchased ACoin in the previous step. The BONUS_GAP(the time interval between two bonus processes) is set small enough so that we can immediately get the bonus result in our test.

For mintFT, mintManyFT, reduceBalance and mintTransferCCoin functions in CCoin, they can only be called by ACoin, we test these functionalities of CCoin by deploying the ACoin smart contract. In the mintManyFT function, we test the cases in which we

can mint a limited number of FTs in one day, and the total balance will increase the corresponding number of values. And if the number of FT mints of one account exceeds 1000, its balance will remain 1000. The test results are shown in Figure 5, all tests are passed.



(a) tests for the ACoin contract



(b) tests for the CCoin contract

Figure 5: tests on the two contracts

5 Challenges during the Project

5.1 Do you need a blockchain

Our initial idea was to use blockchain to record users' medical records similar to [ZMC22], but this ran into problems with GDPR privacy regulations [18]. Following the advice of our teachers and teaching assistants, we followed the process proposed in the [WG18] paper to think that recording copyright information with blockchain should be considered as both needing blockchain and not involving privacy violation.

5.2 Which development path to follow

At the beginning of the project, we do not really understand what a smart contract is. Not even to say what role JavaScript plays. Until now we realize that a smart contract acts like a set of functions living in the cloud, and it does not have a main() function. Meanwhile, JavaScript takes the public key of the users and calls functions of the smart contract on behalf of the users, where some key operations, like payment, will ask users to sign with a private key.

After some searching, we find that there are three development environments, i.e., Remix, Truffle and Hardhat. Remix is good for rapid prototyping since it provides cloud IDE. Truffle Ganache has a graphical interface, and Hardhat is probably the most free, as it uses npm to manage dependencies.

5.3 Debug solidity

It is nearly impossible to write bug-free code. However, since we cannot run solidity files directly, it looks like we cannot set breakpoints. We follow Hardhat's suggestion to use chai and mocha to perform unit tests, which help us find a lot of bugs.

5.4 Frontend development

None of us has much experience in front-end development, but to develop a relatively complete system, we hope to connect our tokens to the popular Wallet app like MetaMask.

6 Discussion

The existence of a bonus mechanism would encourage creators to use our system to register copyright. For companies and people who often need to buy copyrights (e.g. publishers, advertisers, etc.), the opportunity to redeem points without transaction fees saves money and makes them more willing to buy ACoin; for authors, more ACoin buyers leads to a larger market for ACoin transactions, and authors are more willing to use ACoin to record their copyrights. ([Gaw14])

Even though the commission might be viewed as unnecessary, there do not appear to be other ideal ways to collect money to give a bonus if we do not take the commission. Advertising could be a potential way to do that if we operate a centralized website for people to register and trade copyrights.

7 Conclusion

In this project, we explored the application of blockchain in copyright protection and trading and developed a relatively complete application that contains smart contracts and a front-end. Through the front-end website, users can register and verify copyrights, as well as buy and sell copyrights. The part that requires payment is automatically confirmed by calling the user's MetaMask wallet. The core operations are located on the blockchain, and the data is open to all, which also takes advantage of the tamper-evident and open nature of the blockchain.

Currently we have not considered too much about the gas fee, therefore there should be a lot of optimization that can be done in the program to reduce gas consumption ([Mas+21]). There are some other functions that could be added, like copyright could be authorized for a limited time to many users at a time. Overall, we believe that our final realized results largely met our expectations.

We also learned how to use MetaData to describe NFT and found that IPFS should be a better way to store the content represented by NFT at this time, but IPFS storage platforms such as Pinata still have a high barrier to use for end users. We checked the OpenSea developer documentation and found that it seems to be possible to shelf our

own developed NFT token on OpenSea. In this way, it might be possible to use CCoin to maintain a membership system-like mechanism to promote purchase and usage while leveraging NFT trading platforms such as Open Sea.

References

[Gaw14] Annabelle Gawer. "Bridging differing perspectives on technological platforms: Toward an integrative framework". en. In: *Research Policy* 43.7 (Sept. 2014), pp. 1239–1249. ISSN: 00487333. DOI: 10.1016/j.respol.2014.03.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0048733314000456 (visited on 07/13/2022).

[Eur17] European Parliament. Directorate General for Parliamentary Research Services. *How blockchain technology could change our lives: in depth analysis.* en. LU: Publications Office, 2017. URL: https://data.europa.eu/doi/10.2861/926645 (visited on 07/13/2022).

[18] *Art. 17 GDPR - Right to erasure ('right to be forgotten').* en-US. Section: Uncategorized. Nov. 2018. URL: https://gdpr.eu/article-17-right-to-be-forgotten/ (visited on 08/21/2022).

[WG18] Karl Wüst and Arthur Gervais. "Do you Need a Blockchain?" In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT).* June 2018, pp. 45–54. DOI: 10.1109/CVCBT.2018.00011.

[ZO18] Sijia Zhao and Donal O'Mahony. "BMCProtector: A Blockchain and Smart Contract Based Application for Music Copyright Protection". In: *Proceedings of the 2018 International Conference on Blockchain Technology and Application.* ICBTA 2018. New York, NY, USA: Association for Computing Machinery, Dec. 2018, pp. 1–5. ISBN: 978-1-4503-6646-5. DOI: 10.1145/3301403.3301404. URL: https://doi.org/10.1145/3301403.3301404 (visited on 08/21/2022).

[Gui+20] Yuan Guichun et al. "Survey on the application of blockchain in Digital Rights Protection". In: *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI).* Dec. 2020, pp. 183–187. DOI: 10.1109/ICHCI51889.2020.00047.

[Mas+21] Nitima Masla et al. "Reduction in Gas Cost for Blockchain Enabled Smart Contract". In: *2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON).* Sept. 2021, pp. 1–6. DOI: 10.1109/GUCON50781.2021.9573701.

[ZMC22] Zixiong Zhao, Jiaqi Ma, and Chinese Centre for Disease Control and Prevention, Beijing, China. "Application of Blockchain in Trusted Digital Vaccination Certificates". en. In: *China CDC Weekly* 4.6 (2022), pp. 106–110. ISSN: 2096-7071. DOI: 10.46234/ccdcw2022.021. URL: http://weekly.chinacdc.cn/en/article/doi/10.46234/ccdcw2022.021 (visited on 08/11/2022).