Security in the NFT world

Bachelor Thesis

Noé Bayard

from

Bern, Switzerland

Faculty of Science, University of Bern

29. June 2023

Prof. Christian Cachin
Jovana Micic
Cryptology and Data Security Group
Institute of Computer Science
University of Bern, Switzerland

Abstract

In this thesis, a general overview of Non-Fungible-Tokens (NFTs) is given, including the technology around them. The main focus here is on the security issues that come with this still very young technology, and aim to find possible solutions to some of those problems. As well as shed light on some issues with very few immediate possible solutions at the moment. There are many security issues addressed here, but the emphasis is on frontrunning in the NFT ecosystem. Frontrunning will be simulated with an implementation.

Contents

Chapter 1

Introduction

NFTs (Non-Fungible Tokens) are unique, tradable tokens stored on a blockchain. A blockchain consists of linked data records or transactions stored on a decentralized network. With the use of cryptographic protocols, these data records are then protected [24]. These transactions are then mined into blocks by miners solving highly computational cryptographic proof of work challenges [23]. Every account in the network can generally see the contents of the blockchain and can add their own transactions to the blockchain. Once a transaction has been added to the blockchain, it cannot be removed or changed due to the linked property of the blockchain. Every change to a block would also change all subsequent blocks, making blockchains immutable. One of the most notable blockchains, especially when considering NFTs, is the Ethereum blockchain [9], which was the first blockchain used to create NFTs. Transactions between two users are made using so-called smart contracts.
Smart contracts are small programs stored on the blockchain. They were originally designed to make online transactions between two participants faster and verifiable [24]. The contents of a smart contract is publicly available, giving each participant the chance to verify for themselves that the smart contract is indeed fair. Once agreed upon, the smart contract performs the transaction. This makes smart contracts a way for untrusting and decentralized participants to make fair transactions without having to trust the other participants or a third party.
NFTs were first popularized with the launch of the Ethereum blockchain in 2015. Unlike cryptocurrencies NFTs are non-fungible, meaning they are all unique, have different values, and cannot be used interchangeably. An NFT always comes with a verifiable ownership record, which proves someone to be the rightful owner of the NFT. NFT transactions work on smart contracts, ensuring a trustless transfer.
NFTs have gained a lot of popularity and attention in recent years. With this recently increased popularity, NFTs have received a fair amount of criticism for various reasons. This has led to NFTs being somewhat controversial nowadays. Therefore, it is also very important to investigate potential security issues in the NFT ecosystem and identify possible solutions to solve them or at least minimize them.
Every NFT has an associated digital asset it points to, it does so usually through a link stored in the NFT metadata. Metadata is a way to efficiently store properties of an NFT, such as name, description, transaction history, etc. This metadata is usually written in the JSON (JavaScript Object Notation) format. The digital assets mentioned earlier are usually stored outside of the blockchain and include data like images, videos, art, or music. Because of the individuality of NFTs, each NFT has its own ascribed value and can be traded on an NFT marketplace.
Chapter 2 gives an overview of some important related issues regarding NFTs. Chapter 3 focuses on the NFT itself, laying the groundwork for what an NFT is and how it can be used. Before exploring NFT related security issues in Chapter 4.

Chapter 2

Related Work

2.1 NFT Challenges

Due to the NFT system being closely coupled with the underlying blockchain, performance may suffer, as many blockchains don't operate at fast speeds. NFT transactions on the Ethereum blockchain reach about 30 TPS (transactions per second) which isn't necessarily fast considering the number of people using the system at the same time. Especially when more people get involved. Another problem when it comes to those transactions, is the high gas price that has to be paid, every time a transaction is being made. These gas prices can go upwards of 60 USD per transaction. The high price for every transaction hinders NFTs from being adopted more frequently [24].
There are a lot of legal and policy issues concerning NFTs. These might include commodities, cross-border, transactions, KYC (Know Your Customer) data, etc. The lack of any KYC laws might make money laundering easier, as trades on the NFT marketplace can be done anonymously and with much ease. While there are marketplaces where proper identification is required to create an account, such as Coinbase, this is not the case for every marketplace. This allows users to stay anonymous and even create multiple accounts. Every country has its own legal standpoint regarding NFTs, some are very strict and others aren't. Many countries are still implementing regulatory laws for NFTs. Also important to mention here is the taxability of NFTs. This is also country-dependent, though in most countries, NFTs aren't considered taxable property yet. This is a problem as it allows for undetected financial crimes. As a result, the taxability of NFTs is strongly desired [24].
The extensibility of NFTs is another challenge. Let's first look at cross-chain compatibility. There are a lot of different NFT ecosystems, and all of them are completely isolated from each other. This means that NFTs cannot cross over to other networks but instead have to stay on the one they already are. This is not an easy problem to fix, as essential blockchain properties such as the decentralized nature of a blockchain would have to be changed to some degree [24]. Most NFTs use Ethereum as the underlying blockchain, making this not as big of a problem as it might seem. The other problem regarding extensibility has to do with the application of updates on an already existing blockchain [22]. One problem might be that previous protocols are in conflict with new modifications.

2.2 NFT Marketplace Issues

Marketplaces play an important role in the NFT ecosystem. The specific way these marketplaces are designed can lead to certain issues. Many NFT marketplaces still don't make use of proper user authentication. This means that users can create multiple accounts with little effort. Each of which is hard to trace back to an individual person, as the accounts are anonymous. This lack of user authentication makes all sorts of fraudulent behavior much easier for the user. Two-factor authentication is another important feature helping with account safety that is unfortunately not widely used in NFT marketplaces.
Other issues in the NFT marketplaces involve the minting and transfering of tokens. A token contract doesn't have to be tested first. The contracts only have to be submitted to Etherscan to be considered

as verified. This can lead to faulty or malicious token contracts that don't serve their intended purpose. For example, a contract that doesn't transer the NFT after the currency has already been received. Once a token has been minted, it is sometimes possible, depending on the marketplace, to alter the already existing metadata of the NFT. This means that the digital asset associated with the NFT can be altered, as the URL of the asset is stored in the metadata. By changing the URL, the NFT potentially loses a majority of its value. Such an attack can be prevented by disallowing changes to the metadata once the NFT has been minted [23].

When it comes to token bidding, many problems arise when the bidding doesn't take place on the blockchain itself, but instead is held off-chain. This allows for users to be able to place as many bids as they want, without having to pay anything upfront, as there is no gas fee. Since bidding is inexpensive, such marketplaces are susceptible to artificial inflation of the bid volume as well as bid pollution. This is a form of abuse where a large number of bids are placed on an item without the intend of actually purchasing the item. At the end of bidding, the highest bidder usually doesn't have enough funds on their account, causing their bid to fail. These kinds of abuses aren't very lucrative when it comes to on-chain bidding, due to the gas fee, though there are ways to circumvent certain fees. For example the royalty fee, which is a fee each secondary seller has to pay to the original creator of the NFT. This fee is dependent on the selling price of the NFT and can be set by the creator. One way to avoid this fee is to simply sell the NFT on a marketplace where the royalty is not set. This is possible due to marketplaces not sharing royalty information with each other. Another way is to handle the payment of the NFT off-platform and sending the NFT directly to the buyer. Furthermore, the royalty fee can also be abused by the creator. Some marketplaces allow the creator to alter the originally set royalty fee after the first sale. Doing this can lead to the secondary seller paying a higher royalty fee than advertised [23].

2.3 Fraudulent User Behavior

Trading malpractices are ways for users to gain an unfair advantage over the average honest user. Such practices include wash trading, shill bidding, and bid shielding. These will be further discussed below.

2.3.1 Wash Trading

Wash trading is the act of artificially inflating the trading volume for a certain NFT. The buyer and the seller work together in an effort to make the NFT seem more valuable and desirable than it actually is. The buyer is most likely the same user as the seller, they just use a second account. The two accounts then repeatedly send an NFT to each other. This then creates the illusion of demand. The wash trading is only profitable if the NFT is actually sold at the end, and that at a way higher price than the original one. The price has to be at least higher than all of the gas fee and royalty fee that was paid to perform the wash trading. There are ways to detect wash trading, namely by finding repeating cycles in the sales graph. The more accounts that are used for the wash trading, the harder it is to properly detect the malpractice [23].

2.3.2 Shill Bidding

Shill bidding is a common strategy used for auctions where the seller bids for their own item to artificially increase its final price. Honest bidders could potentially pay significantly more than they should have, as a result of this. Shill bidding is used in a lot of sales, unfortunately there isn't an easy way to detect this behavior. Especially when considering the possibility of multiple accounts being used to perform the shill bidding. User accounts can have certain characteristics, making them suspicious of shill bidding. These include very little trading activity, the placing of monotonically increasing bids, never buying the item, and of course when there is a direct connection between the seller and the bidder [23].

2.3.3 Bid Shielding

Lastly, bid shielding is when a very high bid is placed on a previously very low bid to make other bidders lose interest, deterring them from placing further bids. Then, shortly before the transaction would go through, the high bid is retracted, causing the low bid to win the auction at a very small price. Bid shielding is relatively easy to detect. A user places a bid on an item that already has a bid on it, then retracts it. This is considered bid shielding if, after the retraction, there were no further bids placed on the item, and the bid placed before the retracted bid won the item [23].

Chapter 3

NFTs

In this chapter, we'll look at NFTs a little bit more in depth and get to know some of the essential interactions and underlying technologies of NFTs specifically the NFT ecosystem and multiple different blockchains and marketplaces used for NFTs

3.1 What Is An NFT?

NFT stands for non-fungible token. Each of these tokens is stored on a blockchain and has a digital asset associated with them. The actual asset is usually not stored directly on the blockchain due to their usual high data size, instead a URI containing a link is stored on the blockchain. This link points to the digital asset stored on the public internet. The digital asset in question can be a variety of different types of data like a picture, music, text, and many more. Though NFTs are considered ownership records, the ownership doesn't include the associated asset. This means the owner of an NFT only owns the NFT containing a link that points to the asset, not the asset itself.

3.2 NFT Opportunities

NFTs can have many uses besides just being digital collectibles. There are already a couple of games that exist today that use NFTs as a key element in the game, namely CrytpoKitties [7], Cryptocats [6], CryptoPunks [8], Meebits [15], Axie Infinity [1], TradeStars [20], Gods Unchanged [14] and many more. These games attract a lot of investors because, in all of the games, there are unique objects such as weapons, pets, or skins that can be traded with other players. Each of these objects has its own rarity and related value. Because these games use NFTs for those unique items, they become collectibles with an ownership record. This feature can make for a very unique gaming experience.
Furthermore, NFTs can be used as a way to host virtual events. While before, a centralized company had to be trusted when selling and validating the tickets to the event. Due to the decentralized properties of the blockchain and the use of smart contracts, this is no longer necessary. NFT-based tickets can be created that can be bought and used efficiently without having to trust any third parties. So there is no risk of buying fraudulent or invalid tickets. Every ticket is unique, and once a ticket has been purchased, it cannot be resold [24]. Every owner of a ticket can prove that they are, in fact, the rightful owner of this ticket.
NFTs also have a lot of use for artists, allowing them to digitalize their art, making it possible to be the owner of some digital data, which can then be sold. This is due to the ownership record of NFTs. On top of that, the transactions are all trust-free and safe, thanks to the smart contracts and properties of the blockchain. This makes for an easy way to create and safely sell digital data. It is also possible for the creator of the data to set a fixed royalty fee. The creator would then receive this fee every time the digital data is being sold [24], which is a behavior not possible in non-digital collectibles. This is useful because it makes the profit much less reliable on the initial price while also making it much more realistic. Because the profit is now partially based on how desirable the NFT is.

The Metaverse is a shared digital space designed using augmented reality, allowing for all sorts of digital activities. The activities are mostly interactive, such as playing videogames or trading collectibles like NFTs. This virtual world makes use of a decentralized environment provided by a blockchain. The fast growth of the blockchain technology is also the reason why the metaverse has made a lot of progress and gained a lot of popularity in recent years [24].

3.3 NFT Ecosystem

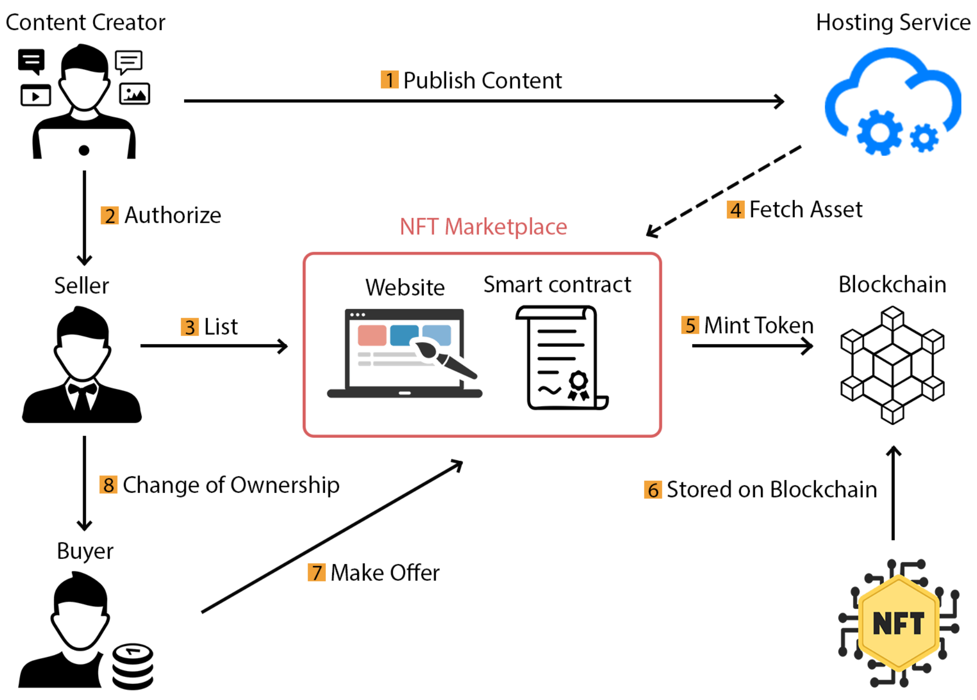

Figure 3.1. Overview of the NFT ecosystem, including all actors in the marketplace as well as all other components

Figure 3.1 depicts the general NFT ecosystem with all its components. The main actors here are the content creator, the seller, and the buyer. The content creator is usually the owner of a digital asset. The seller is the person who mints the NFT with the assigned asset, and the buyer is a person interested in buying the NFT. Other components of the ecosystem include: hosting service, the NFT marketplace, the blockchain, and of course the NFT itself. The hosting service is usually an external website where the digital asset is being stored on. This website is also the one the NFT points to. The NFT marketplace is the place where all NFT transactions happen, it consists of a website that uses smart contracts for its transactions. Let's go through the creation and buying of an NFT. This process starts with the content creator, they start by publishing their digital asset to the hosting service (1), making it publicly available to everyone. They then authorize (2) the seller to list (3) their asset for sale on the NFT marketplace. Then the asset is fetched from the hosting service (4), and finally the NFT can be minted (5) and stored on the blockchain (6). A potential buyer now has to make an offer on the NFT marketplace (7). When accepted, the transaction happens with the help of the smart contract to result in a change of ownership (8). Now the buyer is the rightful owner of the NFT and could list it back on the marketplace [23].

3.4 NFT Blockchain Networks

This is a collection of some popular NFT blockchain networks with a very brief description of each of them. Most of the blockchain networks that support NFTs use the classic token standard ERC-721, containing a tokenid and a few methods to be able to trade tokens and perceive important information such as the current token balance. There also exists the token standard ERC-1155, which has the same functionalities for non-fungible tokens, meaning ERC-1155 can do all the things ERC-721 can. But ERC-1155 also supports fungible tokens [11].

3.4.1 Ethereum

First launched in 2015, Ethereum [9] is the most popular blockchain when it comes to NFTs, with over 90% of NFTs existing on the Ethereum blockchain. Ethereum was the first blockchain to use smart contracts, allowing non-fungible tokens to exist. So it is not surprising to see that Ethereum was the first ever blockchain to be used for creating NFTs. Ethereum declared the ERC-721 standard as the standard used for NFTs and is by far the most notable blockchain for the development of NFTs, having greatly popularized the concept of an NFT. Many NFT projects, such as cryptokitties [7] have pioneered on the Ethereum blockchain. Ethereum has its own native cryptocurrency called Ether, which is one of the most valuable cryptocurrencies today.

3.4.2 Solana

Solana [19] is a competitor to the Ethereum blockchain and considered to be one of the fastest blockchains, because of its hybrid consensus mechanism. This mechanism combines proof-of-history (PoH) and proof-of-stake (PoS) which makes validation time for transactions and smart contract execution faster. Proof-of-Stake means the users themselves validate transactions and get rewarded if they do so correctly. In this consensus model, users have to ”stake” a certain amount of their currency. When a user tries to perform a transaction other users find invalid, the user who added the transaction loses a portion of their stake. This is a way to prevent fraudulent behavior. Proof-of-History ensures historical data is accurate. This is done by creating a hash of past transactions.

3.4.3 Cardano

Cardano [4] is known for being an environmentally friendly blockchain and also one of the fastest growing blockchains. Cardano was created by a co-founder of Ethereum. It has its own native currency, called ADA and features an Ethereum virtual machine, which makes cross-chain compatibility with Ethereum smart contracts possible. The blockchain can be split into two layers: Cardano Settlement Layer (CSL) and Cardano Computation Layer (CCL). CSL is responsible for recording transactions, and CCL hosts the smart contracts.

3.4.4 Binance Smart Chain

Binance Smart Chain [3] is a blockchain based on the Binance chain. It has centralized properties, meaning user security isn’t granted and there is a higher exchange fee. Binance Smart Chain has its own token standard called BEP-721 and is compatible with Ethereum virtual machine. It uses the proof-of-staked-authority consensus model. In this model, users put up a stake to potentially become a validator, who can create blocks and validate transactions.

3.4.5 Flow

Flow [13] is a relatively new blockchain network, having been released in 2019 by the same developers responsible for the blockchain-based game cryptokitties. Despite the blockchain being so new, it is very popular and one of the main alternatives to Ethereum. It is very well known for creating blockchain-based games and dApps. Flow only relies on the proof-of-stake (PoS) consensus model. This consensus model makes Flow very fast compared to most other blockchains.

3.4.6 Ripple

Ripple [18] was founded in 2012. Their blockchain is known to be fast and sustainable. It has a very low average transaction fee of 0.0002 USD because of its very energy efficient validation process. Ripple works on financial inclusion by improving existing infrastructure and creating a sustainable financial system. Ripple has its own cryptocurrency called XRP.

3.5 NFT Marketplaces

3.5.1 OpenSea

Having launched in 2017, OpenSea [16] is one of the oldest NFT marketplaces. It is considered the largest, and one of the most beginner-friendly NFT marketplaces with the highest number of NTF sales. OpenSea has many types of NFTs like art, music, photos, trading cards, and other collectibles [2]. It allows users to mint their own NFTs. With a transfer fee of 2.5% per transaction, OpenSea is on the lower side when it comes to transfer fees. Though there are reportedly many scams on OpenSea, this might be due to the lack of user authentication and the possibility of altering data post-sale [23].

3.5.2 Axie Infinity

Axie Infinity [1] is a marketplace where gaming NFTs can be traded. There are no other NFTs on the marketplace besides gaming NFTs for Axie Infinity, making NFT options very limited. Unlike OpenSea, Axie Infinity only works with the Ethereum blockchain and the Ronin blockchain, ether is the only currency used. Transactions have a fee of 5.25%, though this fee only has to be paid by the seller. The buyer of an NFT doesn't pay any fee [2]. Axie Infinity allows the creator of an NFT to modify the URL in the metadata at any time.

3.5.3 Larva Labs CryptoPunks

Larva Labs CryptoPunks [8] is a very unique marketplace for NFTs in many ways. Firstly, the design differs from other NFT marketplaces. The design features a large collection of characters with different attributes that users can look through. Secondly, it doesn't charge any transactionfees at all. This second point is very rare and usually not the case for other marketplaces. Larva Labs doesn't feature a great variety of NFTs, they are most well known for the CryptoPunks NFT project and also Meebits. Being exclusive to the Ethereum blockchain, ether is the currency used.

Chapter 4

Security Issues

In this chapter, we'll focus on possible security issues concerning NFTs and potential solutions to solve them. The NFT system generally consists of blockchain, storage, and web applications, which all have different security issues and potential flaws. A lot of very well-known security issues are still relevant in the NFT system, these include spoofing, tampering, repudiation, information disclosure, denial of service, and evaluation of privileges. Most of these issues don't affect the blockchain, as it's a naturally very robust and resilient technology. These security issues usually affect the assets stored outside of the blockchain, meaning the web hosting services used for the assets. Another potential target for these attacks can be the web application where the NFT marketplace is located on.

4.1 Frontrunning

Frontrunning the NFT marketplace works a little bit differently from regular frontrunning used in the stock market. In the regular stock market, the frontrunning happens due to insider information concerning potential future purchases. While this type of insider information is also called frontrunning when it comes to NFTs, there is a type of frontrunning that is very unique to NFTs. This is what we'll be focusing on now. An important difference is that the insider information in this case is publicly available to everyone. The information lies in a so-called mempool. The mempool is a place where transactions are being stored that have not been executed or mined by the miner yet. This is before the new block is created. Every transaction on a blockchain costs gas, which is supposed to compensate for the computational work the miner has to perform when validating the transactions in the mempool, usually done with proof of work consensus algorithms. The natural consequence of this is that transactions with a higher gas price are mined first because they are prioritized by the miners. Meaning the order of transactions is determined by the gas price paid when submitting the transaction. The idea of frontrunning is to observe this mempool and see all the future transactions, so all buy requests of individual NFTs. Then the frontrunner attempts to submit their own transaction with a higher gas price before the original transaction was mined. Because of the higher gas price, the miner prioritizes the frontrunner's transaction and performs it first, meaning the frontrunner was able to buy the NFT in question ahead of the user, as seen in figure 4.1. Now the NFT is immediately listed back on the marketplace for a higher price, effectively profiting in the process. In actual practice, the frontrunner is not a real person, but rather a bot that is programmed to frontrun incoming transactions in the mempool, as manually frontrunning would be way too fast for any human to do by themselves [5]. In traditional frontrunning, the most effective countermeasure is to try to obscure the incoming orders as best as possible, making it very hard to gain any inside information, therefore preventing the frontrunning attack. When it comes to NFTs, though, this solution is not possible, as it would require for the mempool to be hidden from the users. This would require a redesign of the blockchain, though hidden mempools already exist. Instead, what can be done in the case of NFT frontrunning is intentionally paying higher gas prices. This is a very simple solution that makes the act of frontrunning much more expensive and therefore less likely. Another solution, which is much harder to implement, would be to create some kind of detection algorithm that scans users' trading data and determines if they are performing frontrunning.

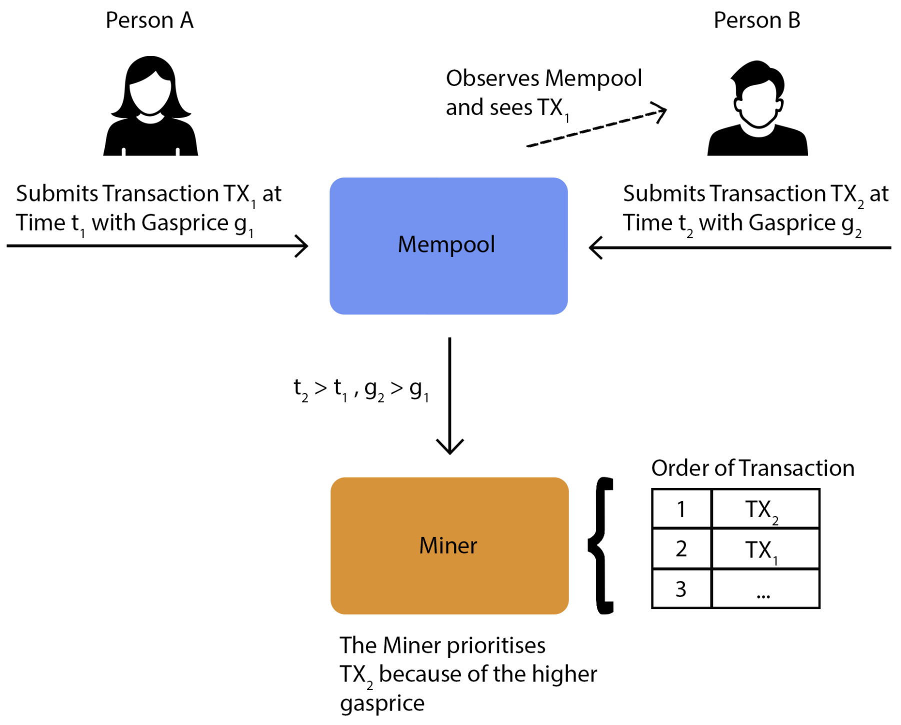
Person A
Person B
Observes Mempool and sees TX_1
Submits Transaction TX_1 at Time t_1 with Gasprice g_1
Mempool
Submits Transaction TX_2 at Time t_2 with Gasprice g_2
$t_2 > t_1$, $g_2 > g_1$
Miner
Order of Transaction
1 TX_2
2 TX_1
3 ...
The Miner prioritises TX_2 because of the higher gasprice

Figure 4.1. Person A submits a transaction. Person B observes the mempool and submits their own transaction with a higher gas price as a response. Due to the higher gas price, the transaction from Person B is mined earlier.

4.1.1 Frontrunning implementation using the XRPL framework

This implementation is based on Ripple's NFT test harness [21]. The test harness uses a test net, which has all the same functionalities as the actual XRP Ledger but doesn't use real currency. The goal of this implementation was to create a simulation of frontrunning in the NFT ecosystem. This happens by using three accounts: a standby account, an operational account, and then the frontrun account. the standby account will first mint an NFT and list it. The operational account will then try to buy the NFT, but gets frontrun by the frontrun account, which will then immediately list it back at a higher price.

```
  function getNet() {
      net = "wss://s.altnet.rippletest.net:51233"
      return net
      }
```

This is a very simple function that only returns the test network.

```
async function getAccount(type) {
    let net = getNet()

    const client = new xrpl.Client(net)
      console.log('Connecting to ' + net + '....')

      // This uses the default faucet for Testnet/Devnet
      let faucetHost = null

      await client.connect()
11
12      console.log('Connected, funding wallet.')
13
14      // -----------------------------------Create and fund a test account wallet
15     const my_wallet = (await client.fundWallet(null, { faucetHost })).wallet
16
17     console.log('Got a wallet.')
18
19      // -----------------------------------Get the current balance.
20      const my_balance = (await client.getXrpBalance(my_wallet.address))
21
22      if (type == 'standby') {
23        standbyAccountField = my_wallet.address
24        standbyPubKeyField = my_wallet.publicKey
25        standbyPrivKeyField = my_wallet.privateKey
26        standbyBalanceField =
27            (await client.getXrpBalance(my_wallet.address))
28        standbySeedField = my_wallet.seed
29        console.log('Standby account created.')
30      } else if (type == 'operational'){
31        operationalAccountField = my_wallet.address
32        operationalPubKeyField = my_wallet.publicKey
33        operationalPrivKeyField = my_wallet.privateKey
34        operationalSeedField = my_wallet.seed
35        operationalBalanceField =
36            (await client.getXrpBalance(my_wallet.address))
37            console.log('Operational account created.')
38      } else {
39        frontrunAccountField = my_wallet.address
40        frontrunPubKeyField = my_wallet.publicKey
41        frontrunPrivKeyField = my_wallet.privateKey
42        frontrunSeedField = my_wallet.seed
43        frontrunBalanceField =
44            (await client.getXrpBalance(my_wallet.address))
45            console.log('Frontrunning account created.')
46      }
47      // --------------- Capture the seeds for both accounts for ease of reload.
48      seeds = standbySeedField + '\n' + operationalSeedField
49      client.disconnect()
50    } // End of getAccount()
```

This creates all of the mentioned accounts and assigns them a personal wallet with 10000 XRP in it. To create the accounts, the function takes in an input telling it what kind of account we are trying to create (1). Then some of the variables of the respective account are assigned with values. These variables include: AccountField, PubKeyField, PrivKeyField, SeedField, BalanceField.

```
async function mintToken() {
  console.log('Connecting to ' + getNet() + '....')
  let net = getNet()
  const standby_wallet = xrpl.Wallet.fromSeed(standbySeedField)
  const client = new xrpl.Client(net)
  await client.connect()
  console.log ('\nConnected. Minting NFToken.')

  // Note that you must convert the token URL to a hexadecimal
  // value for this transaction.
  // ------------------------------------------------------------------------
  const transactionBlob = {
    "TransactionType": "NFTokenMint",
    "Account": standby_wallet.classicAddress,
    "URI": xrpl.convertStringToHex(standbyTokenUrlField),
    "Flags": parseInt(standbyFlagsField),
    "TransferFee": parseInt(standbyTransferFeeField),
    "NFTokenTaxon": 0 //Required, but if you have no use for it, set to zero.
19   }
20
21   // ----------------------------------------------------- Submit signed blob
22   const tx = await client.submitAndWait(transactionBlob, { wallet: standby_wallet} )
23   const nfts = await client.request({
24     method: "account_nfts",
25     account: standby_wallet.classicAddress
26   })
27
28   // ------------------------------------------------------- Report results
29   console.log ('\n\nTransaction result: '+ tx.result.meta.TransactionResult)
30   standbyBalanceField =
31     (await client.getXrpBalance(standby_wallet.address))
32
33   standbyTokenIdField = nfts.result.account_nfts[0].NFTokenID
34
35   client.disconnect()
36 } //End of mintToken()
```

Moving on to the part of the code where the actual NFTs are being minted. As mentioned previously, the standbyaccount is the one creating the NFT, so at first the function accesses the wallet of the standby account through its SeedField (4). Once the wallet has been obtained, a transactionBlob is created (12) with the variables: TransactionType (13), Account (14), URI (15), Flags (16), TransferFee (17), NFTokenTaxon (18). TransactionType is the desired transaction in this case, it's ”NFTokenMint”, account referes to the account minting the NFT, this is the standby account URI, which contains the NFT's URL. This URL is defined in the variables above. Then we have the TransferFee which is also defined above and is exactly what it sounds like, when there is a transaction made with the newly minted NFT, this is the fee that has to be paid. The now created transactionBlob is submitted to the client to then successfully mint the NFT with the chosen attributes. At the end of the function, the tokenIdField of the standby account is filled with the newly minted NFT's token id (33).

```
1  async function getTokens(type) {
2    let wallet
3    if (type == 'standby')
4      wallet = xrpl.Wallet.fromSeed(standbySeedField)
5    else if (type == 'operational')
6      wallet = xrpl.Wallet.fromSeed(operationalSeedField)
7      else wallet = xrpl.Wallet.fromSeed(frontrunSeedField)
8    let net = getNet()
9    const client = new xrpl.Client(net)
10   console.log ('Connecting to ' + getNet() + '...')
11   await client.connect()
12   console.log ('\nConnected. Getting NFTokens...')
13   const nfts = await client.request({
14     method: "account_nfts",
15     account: wallet.classicAddress
16   })
17   console.log ('\nNFTs:\n ' + JSON.stringify(nfts,null,2))
18   client.disconnect()
19 } //End of getTokens()
```

This part of the code is used to gain information about an account. The account in question has to again be given to the function through an input value (1). Then the wallet of the account is accessed with the seedField (4/6), same as in mintToken(). With the correct wallet acquired, the client is then requested to return the account NFTs (13).

```
1  async function createSellOffer(type) {
2    standbyFlagsField = 1
3    const standby_wallet = xrpl.Wallet.fromSeed(standbySeedField)
4    const operational_wallet = xrpl.Wallet.fromSeed(operationalSeedField)
5    const frontrun_wallet = xrpl.Wallet.fromSeed(frontrunSeedField)
6    let net = getNet()
```

```
  const client = new xrpl.Client(net)
  console.log ('Connecting to ' + getNet() + '...')
  await client.connect()
  console.log ('\nConnected. Creating sell offer...')

  let wallet
  let tokenId
  let amount
  if (type == 'standby'){
    wallet = standby_wallet
    tokenId = standbyTokenIdField
    amount = standbyAmountField
  }
  else if (type == 'operational'){
    wallet = operational_wallet
    tokenId = operationalTokenIdField
    amount = operationalAmountField
  }
      else {
        wallet = frontrun_wallet
        tokenId = frontrunTokenIdField
        amount = frontrunAmountField

    }

  // Prepare transaction -------------------------------------------------------
  let transactionBlob = {
    "TransactionType": "NFTokenCreateOffer",
    "Account": wallet.classicAddress,
    "NFTokenID": tokenId,
    "Amount": amount,
    "Flags": parseInt(standbyFlagsField),
  }

// Submit transaction --------------------------------------------------------

  const tx = await client.submitAndWait(transactionBlob,{wallet: wallet})

  console.log ('\n\n***Sell Offers***\n')

  let nftSellOffers
  try {
    nftSellOffers = await client.request({
      method: "nft_sell_offers",
      nft_id: tokenId})

      if (type == 'standby')
          standbyTokenOfferIndexField = nftSellOffers.result.offers[0].
              nft_offer_index
      else if (type == 'operational')
          operationalTokenOfferIndexField = nftSellOffers.result.offers[0].
              nft_offer_index
      else
          frontrunTokenOfferIndexField = nftSellOffers.result.offers[0].
              nft_offer_index

  } catch (err) {
    nftSellOffers = "No sell offers."
  }
  console.log (JSON.stringify(nftSellOffers,null,2))

// Check transaction results -------------------------------------------------
  console.log ('\n\nTransaction result:\n' +
67     JSON.stringify(tx.result.meta.TransactionResult, null, 2))
68   console.log ('\n\nBalance changes:\n' +
69     JSON.stringify(xrpl.getBalanceChanges(tx.result.meta), null, 2))
70   operationalBalanceField =
71     (await client.getXrpBalance(operational_wallet.address))
72   standbyBalanceField =
73     (await client.getXrpBalance(standby_wallet.address))
74   frontrunBalanceField =
75     (await client.getXrpBalance(frontrun_wallet.address))
76
77   client.disconnect()
78 }// End of createSellOffer()
```

Finally, we get to the first part of a transaction, the creation of a sell offer. first, as before, the wallet is accessed (3-5). The variables wallet, tokenId and amount are then assigned the correct values corresponding to the respective account that wants to create the sell offer (12-28). Again, a transactionBlob is creating (33) with the variables: TransactionType (34), Account (35), NFTokenID (36), Amount (37), Flags (38). NFTokenID is the unique identifier of the NFT, this is included to know which NFT of the account we should create the sell offer for. The variable Amount refers to the amount the seller wants to sell it for. This blob is again submitted to the client to perform the action (49). Afterwards, a token offer index is being created and stored in the respective account's variable (54/56/58), to uniquely identify the sell offer that was just created. At the end of this function, the balance field of each account is updated (70-75).

```
1  async function acceptSellOffer(type, seller) {
2    const standby_wallet = xrpl.Wallet.fromSeed(standbySeedField)
3    const operational_wallet = xrpl.Wallet.fromSeed(operationalSeedField)
4    const frontrun_wallet = xrpl.Wallet.fromSeed(frontrunSeedField)
5    let net = getNet()
6    const client = new xrpl.Client(net)
7    console.log ('Connecting to ' + getNet() + '...')
8    await client.connect()
9    console.log ('\nConnected. Accepting sell offer...\n\n')
10
11   let wallet
12   let tokenOfferIndex
13   let tokenId
14
15   if (seller == 'standby'){
16   tokenOfferIndex = standbyTokenOfferIndexField
17   tokenId = standbyTokenIdField
18 }
19   else if (seller == 'operational'){
20   tokenOfferIndex = operationalTokenOfferIndexField
21   tokenId = operationalTokenIdField
22 }
23     else {
24     tokenOfferIndex = frontrunTokenOfferIndexField
25     tokenId = frontrunTokenIdField
26   }
27
28   if (type == 'standby'){
29     wallet = standby_wallet
30     standbyTokenIdField = tokenId
31   }
32   else if (type == 'operational'){
33     wallet = operational_wallet
34     operationalTokenIdField = tokenId
35   }
36     else {
37       wallet = frontrun_wallet
38       frontrunTokenIdField = tokenId
39      }
40  
41    // Prepare transaction ---------------------------------------------------------
42    const transactionBlob = {
43      "TransactionType": "NFTokenAcceptOffer",
44      "Account": wallet.classicAddress,
45      "NFTokenSellOffer": tokenOfferIndex,
46    }
47  
48    // Submit transaction ----------------------------------------------------------
49    const tx = await client.submitAndWait(transactionBlob,{wallet: wallet})
50    const nfts = await client.request({
51      method: "account_nfts",
52      account: wallet.classicAddress })
53  
54    // Check transaction results --------------------------------------------------
55  
56    standbyBalanceField =
57      (await client.getXrpBalance(standby_wallet.address))
58    operationalBalanceField =
59      (await client.getXrpBalance(operational_wallet.address))
60    frontrunBalanceField =
61      (await client.getXrpBalance(frontrun_wallet.address))
62  
63    console.log ('Transaction result:\n')
64    console.log (JSON.stringify(tx.result.meta.TransactionResult, null, 2))
65    console.log ('\nBalance changes:')
66    console.log (JSON.stringify(xrpl.getBalanceChanges(tx.result.meta), null, 2))
67    console.log (JSON.stringify(nfts,null,2))
68  
69    client.disconnect()
70  }// End of acceptSellOffer()
```

This is the second part of the transaction. This function is used to accept the previously created sell offer. It does so by again getting access to the account's wallets (2-4) and assigning variables based on what the accounts in question are (15-38). The variables are wallet, tokenOfferIndex and tokenId. This is where the NFT's token id changed the account it is stored in due to the NFT changing ownership. A transactionBlob is created (42), this time using the value of the tokenOfferIndex (45) to know which sell offer is meant. The blob is submitted to the client (49), finishing the transaction between the chosen accounts. At the end, the balance of the accounts has to be updated again (56-61).

```
  async function placeBid(type, gas){
  const set = [type,gas]
  if (orderOfTransaction.length == 0){
      orderOfTransaction[0] = set
  }
  else for (i=0;i<=orderOfTransaction.length;i++){
      if (orderOfTransaction[i] == null){
          orderOfTransaction.push(set)
          break
         }
         else if (orderOfTransaction[i][1]<gas){
          orderOfTransaction.splice(i,0,set)
          break
         }
  }
}

async function performTransactions(){
  for (i=0;i<orderOfTransaction.length;i++){
    await acceptSellOffer(orderOfTransaction[i][0],seller)
    if (orderOfTransaction[i][0] == 'frontrun')
      await createSellOffer('frontrun')
  }
}
```

These two functions are the logic of the frontrunning. placeBid acts like a mempool where transactions can be stored. Every transaction is then sorted and saved in an array based on the gas price that was given as an input. performTransactions, as the name would suggest, goes through this array and performs all the transactions in the new order. This combination is the reason why the frontrunning account gets to buy the NFT before the operational account, even though its bid was placed later on.

```
  async function main(){
    console.log("Test XPR NFT minting")

    //Get New Standby Account
    await getAccount('standby')
    await getAccount('operational')
    await getAccount('frontrun')
    await mintToken()
    await createSellOffer('standby')

    placeBid('operational',16)
    placeBid('frontrun',20)
    await performTransactions()

    await getTokens('standby')
    await getTokens('operational')
    await getTokens('frontrun')

    console.log("Standby Balance: " + standbyBalanceField)
    console.log("Operational Balance: " + operationalBalanceField)
    console.log("Frontrun Balance: " + frontrunBalanceField)
}
```

Lastly, we have the main function, which is the first function that is called when starting the program. It performs all of the above-mentioned functions in an order that effectively frontruns the operational account. At the end, all the tokens of all accounts are shown, as well as their balance.

```
Test XPR NFT minting
Connecting to wss://s.altnet.rippletest.net:51233....
Connected, funding wallet.
Got a wallet.
Standby account created.
Connecting to wss://s.altnet.rippletest.net:51233....
Connected, funding wallet.
Got a wallet.
Operational account created.
Connecting to wss://s.altnet.rippletest.net:51233....
Connected, funding wallet.
Got a wallet.
Frontrunning account created.
Connecting to wss://s.altnet.rippletest.net:51233....

Connected. Minting NFToken.


Transaction result: tesSUCCESS
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Creating sell offer...


***Sell Offers***
{
  "id": 12,
  "result": {
    "nft_id": "00080014F87385AC1AEC0B6524A0766D92AB2F7AAEE2F0280000099B00000000",
    "offers": [
      {
        "amount": "10000000",
        "flags": 1,
        "nft_offer_index": "
            CE88264AA0F9132E742E459661D1432D87603E1EA3198F9448B65803B47E521C",
        "owner": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG"
      }
    ]
  },
  "type": "response"
}


Transaction result:
"tesSUCCESS"


Balance changes:
[
  {
    "account": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
    "balances": [
      {
        "currency": "XRP",
        "value": "-0.000012"
      }
    ]
  }
]
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Accepting sell offer...
Transaction result:
"tesSUCCESS"

Balance changes:
[
  {
    "account": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
    "balances": [
      {
        "currency": "XRP",
        "value": "10"
      }
    ]
  },
  {
    "account": "rMRrLR42vVaus5jeQsaUGLFRt4yL3VpXUr",
    "balances": [
      {
        "currency": "XRP",
        "value": "-10.000012"
      }
    ]
  }
]
{
  "id": 12,
  "result": {
    "account": "rMRrLR42vVaus5jeQsaUGLFRt4yL3VpXUr",
    "account_nfts": [
      {
        "Flags": 8,
        "Issuer": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
        "NFTokenID": "00080014F87385AC1AEC0B6524A0766D92AB2F7AAEE2F0280000099B00000000",
        "NFTokenTaxon": 0,
        "TransferFee": 20,
        "URI": "697066733A2F2F62616679626569676479727A74357366703775646D37687537367568",
        "nft_serial": 0
      }
    ],
    "ledger_current_index": 38273634,
    "validated": false
  },
  "type": "response"
}
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Creating sell offer...


***Sell Offers***
{
  "id": 14,
  "result": {
    "nft_id": "00080014F87385AC1AEC0B6524A0766D92AB2F7AAEE2F0280000099B00000000",
    "offers": [
      {
        "amount": "11000000",
        "flags": 1,
        "nft_offer_index": "12BA9CBE1E62AAEBF836A1D28AEE9B2AC2BAC2A9E9B759F0BC423C859D324236",
        "owner": "rMRrLR42vVaus5jeQsaUGLFRt4yL3VpXUr"
      }
    ]
  },
  "type": "response"
}


Transaction result:
"tesSUCCESS"


Balance changes:
[
  {
    "account": "rMRrLR42vVaus5jeQsaUGLFRt4yL3VpXUr",
    "balances": [
      {
        "currency": "XRP",
        "value": "-0.000012"
      }
    ]
  }
]
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Accepting sell offer...
Transaction result:
"tesSUCCESS"

```

```
Balance changes:
[
  {
    "account": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
    "balances": [
      {
        "currency": "XRP",
        "value": "0.0022"
      }
    ]
  },
  {
    "account": "rnuTA2htMEhbsNrvWP3hahqm4dttnr2B52",
    "balances": [
      {
        "currency": "XRP",
        "value": "-11.000012"
      }
    ]
  },
  {
    "account": "rMRrLR42vVaus5jeQsaUGLFRt4yL3VpXUr",
    "balances": [
      {
        "currency": "XRP",
        "value": "10.9978"
      }
    ]
  }
]
{
  "id": 12,
  "result": {
    "account": "rnuTA2htMEhbsNrvWP3hahqm4dttnr2B52",
    "account_nfts": [
      {
        "Flags": 8,
        "Issuer": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
        "NFTokenID": "00080014
            F87385AC1AEC0B6524A0766D92AB2F7AAEE2F0280000099B00000000",
        "NFTokenTaxon": 0,
        "TransferFee": 20,
        "URI": "697066733
            A2F2F62616679626569676479727A74357366703775646D37687537367568",
        "nft_serial": 0
      }
    ],
    "ledger_current_index": 38273638,
    "validated": false
  },
  "type": "response"
}
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Getting NFTokens...

NFTs:
 {
  "id": 0,
  "result": {
    "account": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
    "account_nfts": [],
    "ledger_current_index": 38273638,
    "validated": false
  },
  "type": "response"
}
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Getting NFTokens...

NFTs:
 {
  "id": 0,
  "result": {
    "account": "rnuTA2htMEhbsNrvWP3hahqm4dttnr2B52",
    "account_nfts": [
      {
        "Flags": 8,
        "Issuer": "rPegyUG3m1VV2WiDJRq4ta9MURA9Pc3djG",
        "NFTokenID": "00080014
            F87385AC1AEC0B6524A0766D92AB2F7AAEE2F0280000099B00000000",
        "NFTokenTaxon": 0,
        "TransferFee": 20,
        "URI": "697066733
            A2F2F62616679626569676479727A74357366703775646D37687537367568",
        "nft_serial": 0
      }
    ],
    "ledger_current_index": 38273638,
    "validated": false
  },
  "type": "response"
}
Connecting to wss://s.altnet.rippletest.net:51233...

Connected. Getting NFTokens...

NFTs:
 {
  "id": 0,
  "result": {
    "account": "rMRrLR42vVaus5jeQsaUGLFRt4yL3VpXUr",
    "account_nfts": [],
    "ledger_current_index": 38273638,
    "validated": false
  },
  "type": "response"
}
Standby Balance: 10010.002176
Operational Balance: 9988.999988
Frontrun Balance: 10000.997776
```

This is the entire output of the code. In this case, every account starts with 10000 XRP as their starting currency. One NFT is minted and sold for 10 XRP. This transaction is then frontrun by the frontrunning account. At the end, the only account that owns an NFT is the second account, meaning the operational account. The balance of each account shows the potential profit of frontrunning, the frontrunning account has made almost 1 XRP over the transaction.

4.2 Integrity

NFT metadata is usually stored in a JSON file, which is then hashed and stored on a blockchain. Assuming the blockchain and hashing work as intended, it isn't possible for the metadata to be tampered with. At least not without disrupting the entire blockchain. However, the data stored outside of the blockchain, meaning the digital assets that the metadata points to, such as picture files, video, music, etc. These assets can be maliciously manipulated in most cases, causing integrity issues [24]. This can, for example happen when the hosting service where the assets are stored on gets hacked or when their servers crash for some reason. In cases like that, the assets cannot be accessed, or even worse, they can be deleted or changed. A way to ensure integrity in this case is to send both the hash data and the original data when making a transaction [24]. A good solution to this problem is to use some kind of decentralized storage network, such as Filecoin [12]. A decentralized storage network uses multiple independent storage providers to store the users data. These properties make the system tamper-proof, thereby ensuring integrity. Retrievability is also ensured, making sure the users can retrieve their data. This makes decentralized storage networks a great option for off-chain data storage. The URI pointing to the data is also completely unique for each piece of data.

4.3 Duplicate NFTs

In general, everyone can create an NFT, so creating duplicate NFTs is certainly possible. For example, creating an NFT that points to someone else's artwork, which already has its own NFT associated with it, whether that is on the same or a different blockchain. Though doing this is most likely illegal, and probably not a good idea anyway, as buyers can clearly see which of the two NFTs came first and is most likely to be the original. That is if the buyer pays enough attention to realize that the NFT is a fraudulent copy of another NFT. The fraudulent NFT won't have a lot of value associated with it, as long as it's clear that the NFT in question is in fact not the original. Not being able to determine whether an NFT is fraudulent or not is the only dangerous part about these duplicates. For that reason, malicious users will try to make their NFTs seem like original NFTs by making them look very similar to the original, in an attempt to fool the average user. This can, for example, be done using non-ASCII characters in the name that look very similar to real ASCII characters [23]. Unfortunately there isn't really any fix for this behavior, as NFT minting would have to be strongly regulated. The only attempt to help with this issue is to make it harder to create a similar looking NFT by banning certain non-ASCII characters, for example.

4.4 NFT MEV

MEV (maximal extractable value) opportunities are a term traditionally used for cryptocurrencies. An MEV opportunity is when a miner or validator reorders, excludes, or includes specific transactions when producing a new block on the blockchain in order to make some profit. well-known examples of such opportunities include DEX arbitrage, liquidations, and sandwich trading. Searchers are users of the network who analyze blockchain data in hopes of finding a profitable MEV opportunity. These searchers will then pay a lot of gas fees in order to get the transaction to be performed first. All of this is done by bots using complex algorithms to analyze the data and automatically make profitable transactions. Because NFT transactions also use the blockchain in the same way, these techniques can be used for NFTs very similarly. Though it is usually not particularly profitable to do this [10].

Chapter 5

Conclusion

Non-fungible tokens are still in their very early stages of development, but despite that, they are very popular and have a lot of future opportunities for safe and trustless transactions. NFTs have a lot of potential and can be used in a variety of different fields very effectively. But all of that potential also comes with issues and concerns. For NFTs to be even more useful, it is important to take these issues seriously and work towards solutions, which might include well-definded government laws and further development and improvement of the underlying technologies supporting NFTs. While many problems are already being worked on and have solutions, like decentralized storage networks, as a means to ensure integrity in the NFT ecosystem, there are still security issues that go under the radar at the expense of the average honest user. For example, frontrunning, fraudulent NFTs, NFT MEVs etc. It is important to mention that the frontrunning implementation isn't a perfect representation of frontrunning, as it uses an isolated test net. It is merely used to further illustrate the act of frontrunning and the potential threat that comes with it.

Bibliography

[1] “axieinfinity, howpublished = https://axieinfinity.com/,.”

[2] “Best NFT Marketplaces, howpublished = https://www.fool.com/the-ascent/cryptocurrency/nft-marketplaces,.”

[3] “bnbchain, howpublished = https://www.bnbchain.org/en/smartchain,.”

[4] “cardano, howpublished = https://cardano.org/,.”

[5] “cointelegraph, howpublished = https://cointelegraph.com/explained/what-is-front-running-in-crypto-and-nft-trading,.”

[6] “cryptocats, howpublished = https://cryptocats.thetwentysix.io/,.”

[7] “cryptokitties, howpublished = https://www.cryptokitties.co/,.”

[8] “cryptopunks, howpublished = https://www.larvalabs.com/cryptopunks,.”

[9] “ethereum, howpublished = https://ethereum.org/en/,.”

[10] “ethereum/mev, howpublished = https://ethereum.org/en/developers/docs/mev/#mev-examples-nfts,.”

[11] “ethereum/standards, howpublished = https://ethereum.org/de/developers/docs/standards/tokens/erc-1155/,.”

[12] “filecoin, howpublished = https://filecoin.io/,.”

[13] “flow, howpublished = https://flow.com/,.”

[14] “godsunchained, howpublished = https://godsunchained.com/,.”

[15] “meebits, howpublished = https://meebits.larvalabs.com/,.”

[16] “OpenSea, howpublished = https://opensea.io/,.”

[17] “originstamp, howpublished = https://originstamp.com/blog/top-5-blockchains-for-nft-development/,.”

[18] “ripple, howpublished = https://ripple.com/xrp/,.”

[19] “solana, howpublished = https://solana.com/de,.”

[20] “tradestars, howpublished = https://tradestars.app/,.”

[21] “XRPL Test harness, howpublished = https://learn.xrpl.org/course/code-with-the-xrpl/,.”

[22] M. Ciampi, N. Karayannidis, A. Kiayias, and D. Zindros, “Updatable blockchains,” in *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II* (L. Chen, N. Li, K. Liang, and S. A. Schneider, eds.), vol. 12309 of *Lecture Notes in Computer Science*, pp. 590–609, Springer, 2020.

[23] D. Das, P. Bose, N. Ruaro, C. Kruegel, and G. Vigna, “Understanding security issues in the NFT ecosystem,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022* (H. Yin, A. Stavrou, C. Cremers, and E. Shi, eds.), pp. 667–681, ACM, 2022.

[24] Q. Wang, R. Li, Q. Wang, and S. Chen, “Non-fungible token (NFT): overview, evaluation, opportunities and challenges,” *CoRR*, vol. abs/2105.07447, 2021.

Erklärung

Erklärung gemäss Art. 30 RSL Phil.-nat. 18

Ich erkläre hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Universität Bern das Recht, die dazu erforderlichen Personendaten zu bearbeiten und Nutzungshandlungen vorzunehmen, insbesondere die schriftliche Arbeit zu vervielfältigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.

29.06.2023	[illegible]
Ort/Datum	Unterschrift