

Anonymizing NFT Transactions

Final Project Report for CS 397/497: Data Privacy

Lucas Cojot
Northwestern University

Patrick Dwyer
Northwestern University

Gilberto Guadiana
Northwestern University

lcojot@u.northwestern.edu

patrickdwyer2023@u.northwestern.edu

gilbertoguadiana2022@u.northwestern.edu

ABSTRACT

The goal of this project was to design a system which can preserve the privacy of NFT owners by obfuscating Ethereum transactions. We achieved this goal by first researching the weaknesses in the Ethereum ecosystem which make it possible for attackers to deanonymize users and attempt phishing attacks. We then designed a system and deployed the tool as a set of smart contracts on the Ethereum test net. The goal of this system design is to allow for people to transact NFTs for Ethereum through a transaction pool all while keeping the price of purchase and the recipient of the NFT hidden from onlookers. Accordingly, we designed a system that randomly selects a price within a range, for each NFT in a batch of NFTs, and then sends each NFT to a randomly selected recipient from a batch of recipients. As a result of using our tool, onlookers cannot be certain of the NFT's exact value, which makes it much less worthwhile for them to attempt to scam the wallet owner. In addition, our system allows NFT buyers to reduce their risk of demonization in a simpler and safer way than the existing, and widely unknown, method of mixing one's Ethereum pre-NFT purchase. In addition, this method only works if either there is at least one other user who's interacted with the Tornado Cash contract the same amount of times in the recent past or the purchaser mixes more Ethereum than they purchase the NFT for.

1 Introduction

The Ethereum blockchain attracts a high number of scammers due to its intrinsic transparency and decentralized nature. One of the most targeted assets are NFTs, especially since their explosive growth in popularity and value in the recent past. The latest major attack was carried out on OpeanSea users [1], the biggest NFT marketplace on the Ethereum blockchain, and targeted a multitude of wealthy users. The attackers were able to steal a total of \$1.7 million dollars worth of NFTs. As technology and more specifically the web is advancing towards a transparent and decentralized future, the need for privacy

and anonymity will grow. We are already seeing growth in this domain through decentralized protocols such as TornadoCash [2] and ETHmixer [3].

In order to preserve privacy, Web3 developers have designed smart contracts which can obfuscate the link between a wallet address and the assets it owns. However, such tools currently only exist for divisible assets such as cryptocurrencies as they take advantage of their divisibility and fungibility.

The goal of this project is to design a system which can preserve the privacy of NFT owners. NFTs are indivisible (non-fungible) assets and can therefore not be used by already existing system designs.

Through our research on the vulnerabilities of the Ethereum ecosystem, we have concluded that the best way to achieve our goals is to hide the value of the NFT within a range as well as the recipient of the NFT such that when attackers see the transaction on the public ledger, they only know that an NFT has been sent to a wallet but have limited information related to its value and owner.

1.2 Methodology

Our first step was to research privacy risks on Ethereum related to NFTs and visualize an attacker's workflow when attempting to scam a user. Because Ethereum is a pseudo-anonymous platform, deanonymization attacks are common and even done by governmental organizations such as the IRS in order to link wallet addresses to individuals or organizations. For NFT scammers, this is usually done by tracking a person's interactions on social networks. For example, NFT owners often make announcements of which NFTs they've purchased on networks like Twitter or Discord. Once this information is public, the attacker needs only to find the latest transaction for this NFT on the public ledger and successfully link that address to the owner of the social media account. After that, the attacker simply needs to send phishing attacks to that

person on social media or email and hope they take the bait.

The next step was to research and thoroughly understand the mechanics behind the decentralized privacy protocols previously mentioned. After going through the codebase of both TornadoCash and ETHmixer, we decided to base our system design on the one of TornadoCash. This protocol is used for people who want to make their cryptocurrency untraceable back to them. Since every atomic ETH (10^{-18} th of an ETH) has a unique ID, much like dollar bills have a serial number, users want to make sure their crypto is untraceable. TornadoCash achieves this by letting multiple users deposit an equal amount of ETH into a transaction pool. The pool then randomly separates and shuffles each atomic ETH and randomly recombines them into each user's initial deposit amount. These amounts are then redistributed to each user who deposited ETH into the pool. Our next goal was for each of us to become well versed in Solidity, Ethereum's native programming language for smart contracts.

Finally, we entered the design phase in which we drafted and implemented our system design to be compatible with NFTs and tested it on the Ethereum test net until it achieved our desired goals.

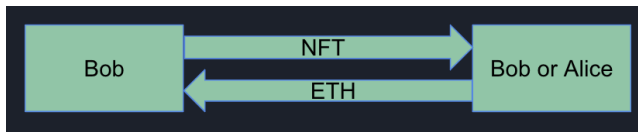


Figure 1: **Anonymous NFT transaction diagram.**

Following the research stages, the following design goals were made for the tool for an anonymous NFT transaction, see Figure 1. Let Bob be the seller of an NFT and Alice be the buyer. We define an NFT transaction to be anonymized if the price of the NFT is obfuscated and the identity of the purchaser can be that of the seller (Alice can be Bob). The first goal is set to lessen the degree of phishing attacks that can occur for the NFT. The second goal allows for a user to do NFT mixing.

1.3 System Design

To meet the design goals of this project, the team designed around a specific context for NFT transactions because trade-offs between utility and privacy were prevalent while doing the system design. This section outlines the context for this tool, the design overview, and the phase by phase design.

1.3.1 System Design Context. To preserve full privacy, as defined in the design goals of this project, of NFT transactions, at least one aspect of the NFTs must be made

common. We chose to limit the NFTs in our system to a predefined range of prices to set a commonality. In this context, the price of any NFT is known by any observer to be within the start and end price values that are publicly available, but observers will not know the exact price of the NFTs because a random function decides the final price. To preserve the privacy of the buyer, we restrict the NFT purchased to be random within the set of NFTs that are for sale. Additionally, the number of sales must be known beforehand to ensure that buyers are able to pay only the final sales price of the NFT. Although the context is restrictive, there are use cases in the NFT ecosystem today.

NFT drops and charities, organizations, or decentralized autonomous organizations that want to raise money can operate within this context. In NFT drops, users receive random NFTs at varying price points. For raising money, NFTs are often sold as part of collections, where each NFT in the collection is worth roughly the same amount.

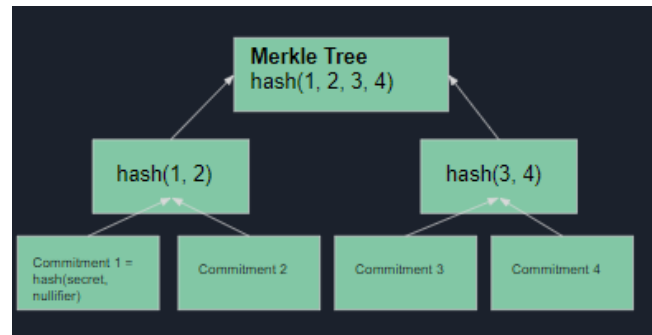


Figure 2: **Commitment merkle tree in Tornado Cash.**

1.3.2 System Design Overview. The tool takes its inspiration from the Ethereum-based Tornado Cash system and uses this system as the starter code [4]. The system uses zero-knowledge proofs, like Tornado Cash. The system separates buyer and seller commitments into two merkle trees, see Figure 2, to allow the buyer and seller to anonymously access buyer and seller functions without revealing their identity. The system uses phases because the design relies on sequential ordering of actions by buyers and sellers. All code is available at [5].

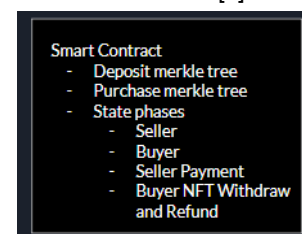


Figure 3: **Additional initialization fields.**

1.3.3 System Design: Initialization. When constructing the smart contract, the system is initialized similarly to how Tornado Cash is initialized. The setup for the zero-knowledge proofs initializes a verifier smart contract and hashing contract. The two merkle trees are constructed and stored, see Figure 3. Additionally, a number of sales (N) and the start and end of the price range is set publicly. Once the constructor is finished, the phase is set to the seller phase.

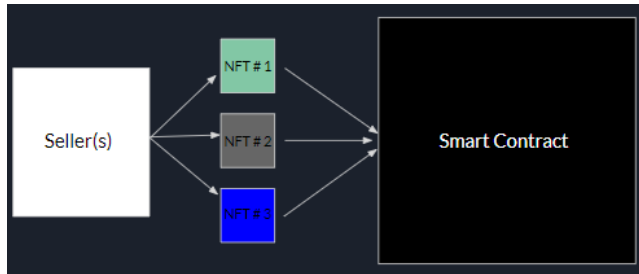


Figure 4: **Seller phase overview.**

1.3.4 System Design: Seller Phase. During the seller phase, the smart contract accepts N NFT deposits. While calling the deposit function, the user must first approve the smart contract address to manage the NFTs that will be deposited. Once deposited, the smart contract adds the commitment to the deposit merkle tree, lists the deposited NFT, and the current phase is set to the buyer phase. The NFTs are now held by the smart contract as shown above.

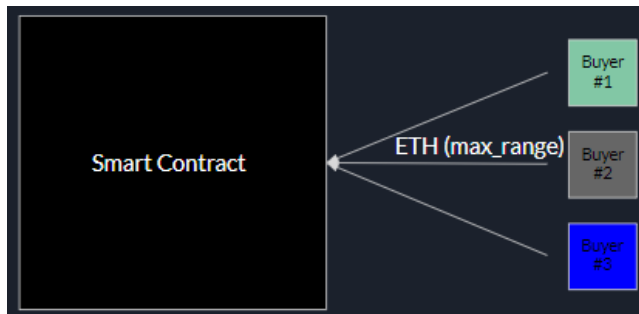


Figure 5: **Buyer phase overview.**

1.3.5 System Design: Buyer Phase. In this phase, N NFT purchases are required to move onto the Seller Payment phase. Sellers send the maximum price of the NFT with their call to the purchase function. The smart contract then adds the commitment to the purchase merkle tree, and the current phase is updated.

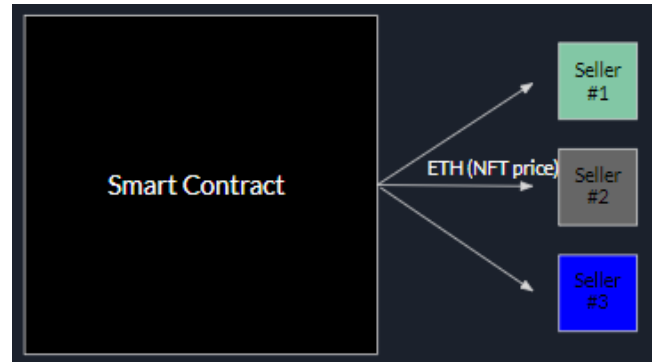


Figure 6: **Seller payment overview.**

1.3.6 System Design: Seller Payment Phase. In this phase, the sales price of the NFTs are determined as the seller receives payment. A random function settles the final sales price of the NFT and publicly posts the price to a list of final sales prices. Once determined, ETH is sent to the seller after the seller uses their deposit commitment to prove that they sold an NFT. For privacy to be preserved, the address that the seller uses to deposit and withdraw payment must be different. Otherwise, the sales price of the NFT is public information. Once all sellers have withdrawn their payment, the phase is updated to the following phase.

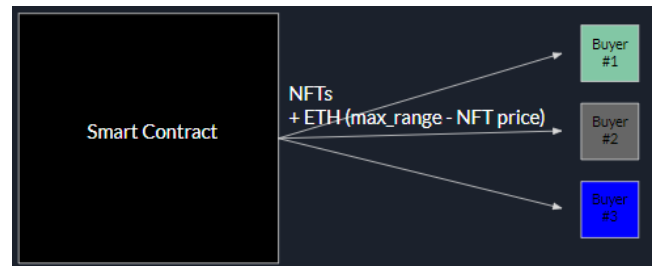


Figure 7: **Buyer NFT Withdraw and Refund Phase overview.**

1.3.7 System Design: Buyer NFT Withdraw and Refund Phase. In this phase, the buyer withdraws their randomly chosen NFT and their refund. A random function selects randomly from the list of deposited NFTs, and a random sales price from the list generated from the previous phase is used to determine the amount to refund the buyer. To not leak information about the price of the sale, the buyer must use different addresses for the NFT and refund. Once all withdrawals are done, the phase is set to the seller phase to restart the cycle.

1.4 System Evaluation

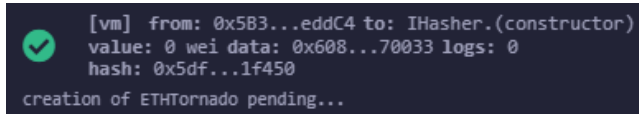


Figure 8: **Successful smart contract deployment to local Ethereum Virtual Machine using Remix IDE.**

To test the functionality of the designed system, the design from the previous section was implemented in Solidity and compiled using the Remix IDE with Solidity compiler version 0.8.7 with optimization enabled (value set to 200). Using Remix, the smart contract was deployed to a local Ethereum Virtual Machine, as shown in Figure 8.

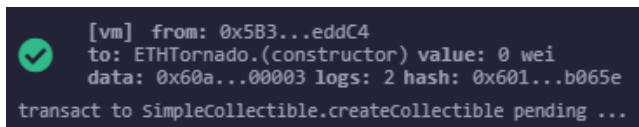


Figure 9: **Successful creation of an NFT.**

To test the smart contract, an NFT generating smart contract was compiled and deployed to create the NFTs that were deposited, as shown in Figure 9 and the contract was used to approve the NFTs to be managed by the privacy smart contract as shown in Figure 10.

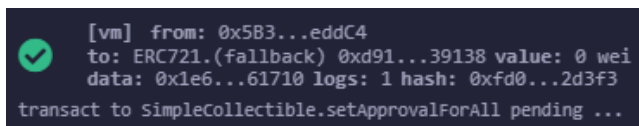


Figure 10: **Approval set for NFTs.**

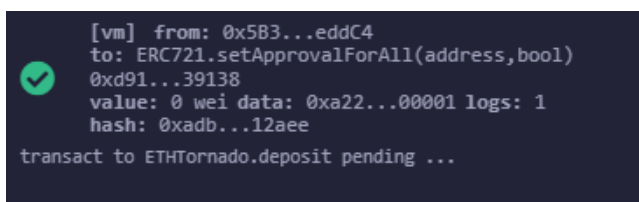


Figure 11: **Successful deposit of an NFT.**

The deposit function was then tested and was successful as shown in Figure 11. A test was done to check whether the purchase function would reject purchases of the wrong amount and this was successful as shown in Figure 12. The purchase function was then tested and was successful as shown in Figure 13.

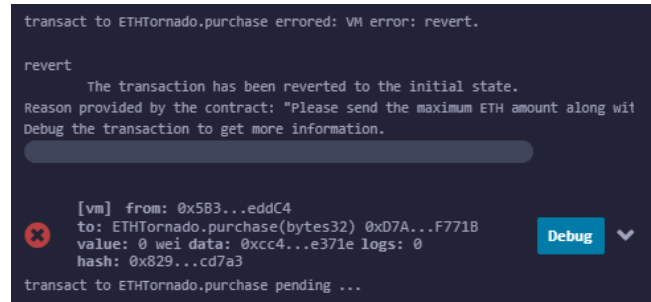


Figure 12: **Rejection of incorrect payment amount.**

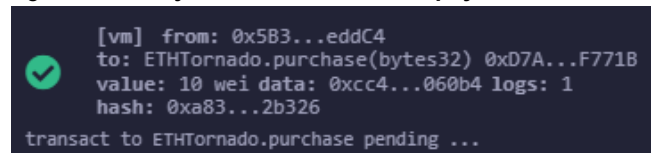


Figure 13: **Successful purchase of an NFT.**

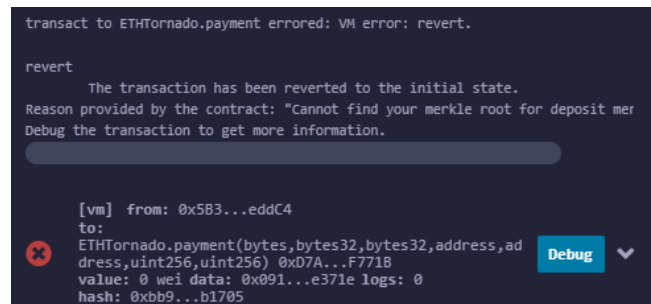


Figure 14: **Rejection of payment withdrawal with incorrect deposit merkle tree root.**

Finally, the payment function was tested to check whether the function would reject proofs with the wrong deposit merkle tree root. That concludes our testing, and all the functions tested are fully functional and bug free.

1.5 Limitations

In general, our team lacked substantial prior knowledge as it relates to the development of smart contracts. In particular, this project required developing a foundational understanding of cryptography and cybersecurity. A vast amount of time was spent suggesting system designs, communally hashing them out and establishing their validity, only to realize some time after that they were flawed in some way. Tornado Cash was the only rigorous and applicable implementation that we were able to develop our understanding from. This may suggest a fundamental limitation in terms of preserving pseudo-anonymity on a public blockchain. Specifically, it is possible that there is not a reasonable way to disconnect the pseudo-identity of a buyer and seller on a public blockchain without hiding transactions in a “crowd”, as Tornado Cash and our system

work. This claim however is largely unfounded and should be expanded on in further research.

Hiding in a “crowd”, which, in this case, means obfuscating the sender and recipient of a digital asset on a public blockchain by making a batch of senders send their asset to a randomly selected recipient from an equally sized batch of recipients, is not without problems, especially when it comes to NFTs. While the assets involved in this batch of transactions may differ, they must be worth the same to each recipient, or the recipients must be willing to gamble their resulting utility. Effectively, we have ensured pseudo-anonymity by treating non-fungible digital assets as if they are fungible (interchangeable).

1.6 Discussion

While our system sacrifices utility for privacy in some respects, there are still many possible use cases for our system. Specifically, and of particular interest, is its potential application to NFT drops, events where a collection of NFTs is distributed randomly (or more likely each NFT is generated randomly on the spot) to participants on a first come first serve basis. Traditionally, these systems are designed for simplicity within relevant smart contracts to decrease gas fees by decreasing needed computation. Accordingly, there are minimal efforts to preserve privacy with respect to purchasing an NFT, and as a result, many utilizers of these NFT marketplaces are subject to highly sophisticated phishing attacks. Even though users may be able to create a new wallet that is theoretically not traceable to their main wallet, they must somehow endow that wallet with money to purchase any NFTs, and this process creates a link between the created wallet and previous wallets. A current potentially viable solution is thus to mix your Ethereum using a system like Tornado Cash, which would allow you to send Ethereum to your new wallet without linking it to your main wallet, and proceed to purchase your NFT. This process however is not widely known and is fairly complicated. Furthermore, if there is not enough traffic on Tornado Cash this method risks de-anonymizing the user. Our system allows a safer and simpler process to be directly integrated into NFT marketplaces.

In addition, our system, unlike the aforementioned Tornado Cash solution, would hide the price that the NFT is sold for within a range. In the NFT space, creators of NFT collections specify an initial price for their NFTs, and this has consequences on the public’s interpretation of the NFTs value. This can lead to malicious schemes where groups of users with significant funds agree to buy many NFTs within a particular project, causing them to sell out when they otherwise may not have.[6] They may also do this post mint (the initial drop of the NFTs). Either way, this inflates the value of the NFTs and causes unsuspecting users to purchase the NFTs at a price higher than the “true” market price. Then, the malicious actors sell their NFTs and leave the market, significantly reducing the overall demand for the NFTs causing their price to drop, leaving users with a

significantly depreciated NFT. Our system, in the hands of project creators who believe in the long-term appreciation of their NFTs, would allow them to sell their NFTs so that onlookers only know that they were sold within a certain range. If this range is large enough, this makes it significantly harder to profit off these pump and dump schemes, which often operate on fairly tight margins.

Notably, this would only apply to the initial distribution of the NFTs, so once the NFTs are on the secondary market they would still be viable to pump and dump attacks. This provides a potential focus for future work as it relates to preserving privacy in the NFT space. We imagine there are many other ways to apply the concepts utilized in our system, and we leave these applications to the imaginations of future researchers and developers, as they are theoretically limitless.

1.7 Conclusion

This is the first system that has attempted to anonymize NFT transactions on the Ethereum blockchain and most likely on any blockchain. What we have produced is a solid foundation for further exploration. Tornado Cash was not the first system of its kind, and our system will most likely never gain the traffic that Tornado Cash has, especially as we would have to pay gas fees to deploy it on Ethereum’s main net. That being said, we are not blockchain developers, so while we are all interested in blockchain technology, this project has posed a steep learning curve that produced more setbacks than anticipated. Particularly, aspects of cybersecurity and cryptography were very difficult and made fully testing some of the later functions in our system infeasible within the allotted time frame. Despite these limitations, we hope that this system provides a backbone for future developers and researchers in the NFT space.

REFERENCES

- [1] L. Olinga, “Scammers steal \$1.7 million worth of nfts from OpenSea customers,” *TheStreet*, 20-Feb-2022. [Online]. Available: <https://www.thestreet.com/investing/cryptocurrency/scammers-steal-1-7-million-worth-of-nfts-from-opensea-customers>. [Accessed: 16-Mar-2022].
- [2] *Tornado.cash*. [Online]. Available: <https://tornado.cash/>. [Accessed: 16-Mar-2022].
- [3] “ETH-Mixer.com,” *Ethereum Mixer | Mix your Ether now! Low fees. Safe. Fast and anonymous.* [Online]. Available: <https://eth-mixer.com/>. [Accessed: 16-Mar-2022].
- [4] TornadoCash, “Tornado-core/contracts at master · tornadocash/tornado-core,” *GitHub*. [Online]. Available: <https://github.com/tornadocash/tornado-core/tree/master/contracts>. [Accessed: 16-Mar-2022].
- [5] “Anonymizing NFTs Repository,” *GitHub*. [Online]. Available: <https://github.com/gguadiana100/anonymizing-NFTs>. [Accessed: 16-Mar-2022].
- [6] D. Scher, “NFT schemes, Scams, & Rugs: The insides of a NFT Pump & Dump GroupD,” *Caught In 4K*, 16-Oct-2021. [Online]. Available: <https://caughtin4k.org/nft-cashgrab-ring-at-large/>. [Accessed: 16-Mar-2022].