NFT Trades in Bitcoin with Off-chain Receipts

Mehmet Sabir Kiraz[1], Enrique Larraia[2], and Owen Vaughan[2]

[1] De Montfort University, UK `mehmet.kiraz@dmu.ac.uk`
[2] nChain, UK `{e.larraia,o.vaughan}@nchain.com`

Abstract. Non-fungible tokens (NFTs) are digital representations of assets stored on a blockchain. It allows content creators to certify authenticity of their digital assets and transfer ownership in a transparent and decentralized way. Popular choices of NFT marketplaces infrastructure include blockchains with smart contract functionality or layer-2 solutions. Surprisingly, researchers have largely avoided building NFT schemes over Bitcoin-like blockchains, most likely due to high transaction fees in the BTC network and the belief that Bitcoin lacks enough programmability to implement fair exchanges. In this work we fill this gap. We propose an NFT scheme where trades are settled in a single Bitcoin transaction as opposed to executing complex smart contracts. We use zero-knowledge proofs (concretely, recursive SNARKs) to prove that two Bitcoin transactions, the issuance transaction $\mathtt{tx}_0$ and the current trade transaction $\mathtt{tx}_n$, are linked through a unique chain of transactions. Indeed, these proofs function as "off-chain receipts" of ownership that can be transferred from the current owner to the new owner using an insecure channel. The size of the proof receipt is short, independent of the total current number of trades n, and can be updated incrementally by anyone at anytime. Marketplaces typically require some degree of token ownership delegation, e.g., escrow accounts, to execute the trade between sellers and buyers that are not online concurrently, and to alleviate transaction fees they resort to off-chain trades. This raises concerns on the transparency and purportedly honest behaviour of marketplaces. We achieve *fair* and *non-custodial* trades by leveraging our off-chain receipts and letting the involved parties carefully sign the trade transaction with appropriate combinations of `sighash` flags.

Keywords: Blockchain · Bitcoin · Zero-knowledge proofs · NFT tokens

1 Introduction

Non-fungible Tokens (NFTs) are digital representations of assets providing ownership records stored on a blockchain. They are cryptographic assets on a blockchain with unique identification codes and pointers to associated metadata, possibly stored off-chain. They can be seen as digital passports or a conventional proof-of-purchase of a digital asset, functioning in the same manner than paper invoices. Blockchains allow two mutually distrustful parties to exchange token ownership for cryptocurrency without a trusted intermediary. The exchange is usually atomic in the sense that it happens in the same transaction or is controlled by a smart contract. NFTs have received huge interest due to their transparency in transaction details, public verifiability and trustless transfer [32]. Unlike conventional systems, public verifiability and transfer of ownership can be tracked continuously [31,3,27,33]. Despite the fact that the complete capabilities of NFTs have not yet been fully realized, they are already being utilized in various business models, including decentralized gaming [18] or e-commerce [7].

NFTs cannot be traded or exchanged in the same way as fungible tokens. Their size can be dynamically large and they cannot be replicated due to its non-fungibility. In Ethereum, the ERC-721 standard [16] defines a minimal interface interface to exchange NFTs. It specifies ownership details, security, and metadata. The ERC-1155 standard [29] builds on top improving several aspects, such as reducing transaction and storage costs by up to 90%, and batching multiple types of NFTs into a single contract. On a separate note, platforms that incorporate off-chain infrastructure enable the listing of tokens, the creation of new tokens (minting), the connection of sellers with buyers, and on-chain settlement of the exchange. NFT marketplaces (NFTM) like OpenSea, Rarible, SuperRare, Foundation, or Nifty, fill this gap and sit between the end users and the blockchain.

NFT security. Token authenticity is the main concern with NFTs. After all, digital assets can easily be copied in fraudulent tokens. In all existing solutions, the burden of verifying the authenticity of the purchased token lies on the buyer. However, as blockchain technology evolves, it becomes harder for regular users (without specialized hardware) to access the network and check by themselves the on-chain state. NFTMs facilitate the trades, but they can impose his view of the blockchain, unilaterally remove tokens from their listings, and tend to rely on off-chain centralized databases that can serve arbitrary content [23]. This questions the purported decentralization and transparency of such platforms. In a recent study [15] a large number of potential vulnerabilities related to NFTMs were identified. We highlight some of the identified issues.

- *Buggy token contracts.* Code of custom contracts deployed by users are not properly audited. They may not pay the seller after the trade, or do not transfer the ownership at all (non-atomicity).
- *Lack of transparency in trading.* For example, Nifty uses escrow accounts and off-chain trades to reduce the transaction fees.
- *Control delegation.* The NFTM takes control of the token and funds to execute the trade without interaction between sellers and buyers. If the NFTM is given full control, it introduces a single point of failure in the security of all its users. Nifty, Foundation and SuperRare follow this approach via escrow accounts. OpenSea has a somewhat safer delegation mechanism, requiring authorization from the seller.
- *Royalty evasion.* If enforcement of royalties fees is delegated to NFTMs, buyers can avoid paying the fee by trading in plattforms that do not set royalties. Also, on-chain standards like ERC-721 do not capture royalties

neither. Hence, payments can be settled off-chain, and transfer ownership on-chain (assuming the parties are willing to conduct non-atomic exchanges).

– *Wash trading.* Sales volume is artificially inflated to create an illusion of demand or to inflate financial metrics of their interests. For example, in Rarible, the more a user spends, the more \$RARI tokens it receives. It is suspected that high-value NFTs such as CryptoKitties are example of wash trading.

Related Work. The authors in [28] proposed a solution called zero-knowledge Address Abstraction (zkAA) to eliminate the need for mapping and enables the direct use of web2 identity in the blockchain. In this way, users can utilize their web2 identities on blockchain through smart contracts leveraging zero-knowledge proofs without disclosing their identity.

In [20], the authors presented an auction protocol for NFTs with supports in multi-chain platforms. The design uses hash time lock transactions and additional strategies to control users malicious behaviour. However, without supporting trustless bridges the multichain platform (i.e., the cross-chain asset exchange process) could be secured. The authors in [12] presented a multi-stage NFT transaction protocol, called LiftChain, which builds a NFT transaction protocol performing a batch of NFT transactions off-chain and then propagates them on-chain. Although it aims to optimize the cost, the existing interfaces as a whole are still very expensive and not scalable.

More related to our work, Ordinal Inscriptions [26,17] is a numbering system that allows individual satoshis (the smallest Bitcoin denomination), referred to as sequence numbers, to be tracked and transferred in ownership. In short, Ordinals can be considered digital assets inscribed in satoshis. More concretely, satoshis are numbered in the order they are mined and transferred based on the first-in, first-out principle of transaction inputs and outputs. Proofs can be created to show that a specific satoshi is indeed present in a specific output. However, these proofs are large, consisting of the block headers and the Merkle path to the coinbase transaction that creates the satoshi, and every transaction that spends the satoshi. Note that the proof size is linear in the number of total current spends, and it increases with each spend. Unlike NFTs that can be purchased through platforms like OpenSea and Nifty Gateway, there are currently no marketplaces or wallets dedicated to Ordinals. They can be traded through Telegram and Discord channels, and the order book is currently maintained as a Google sheet.

Our contributions and techniques. Researchers generally refrained from creating and sending NFTs over the BTC network due to its expensive transaction fees and purported lack of programmability. In this paper, we fill this gap by explaining how to achieve fair exchanges without control delegation on Bitcoin-like blockchains with few transaction fees. More specifically, our main contributions are as follows.

- We design, implement, and benchmark, a proof system to prove and verify that an issuance transaction tx_0 (with an embedded representation of the token), and the current trade transaction tx_n are linked through a unique transaction chain. Our proof system is a recursive SNARK [14,5,2,10,11,22]. This means that if an additional transaction tx_{n+1} is added to the chain, then the proof π_n can be updated into π_{n+1} incrementally, without requiring all previous transactions. The proof receipt π_n functions as an '*off-chain receipt*' handed by the current owner to the new owner of the token. Since we use SNARKs, the size of π_n is constant or just logarithmic in the number n of total current trades.
- We present a new scheme $\mathsf{NFT} = (\mathsf{mint}, \mathsf{list}, \mathsf{sell}, \mathsf{buy})$ to trade NFTs on Bitcoin-like blockchains in the presence of untrusted marketplaces. A trade is settled in just one transaction tx_n and hence it is atomic: the payment and the ownership transfer cannot be decoupled. We guarantee token authenticity leveraging the off-chain proof receipt π_n to dispense access to the blockchain or the need of trusted intermediaries. To provide fairness for sellers and buyers without delegating control of tokens or funds to intermediaries, we employ appropriate combinations of `sighash` flags when signing (unlocking) inputs of tx_n.

In our NFT scheme, all trades appear on-chain, therefore it is transparent by design. As mentioned, the trade is settled by publishing a single transaction on-chain, as opposed to deploying complex smart contracts. The off-chain mechanism for token authenticity is also very flexible and can be enhanced in a number of ways. For example, the issuer can enforce royalty fees, or wash trading can be mitigated. Last, it is worth mentioning that although we describe our NFT scheme for Bitcoin, our techniques are fundamentally independent of the underlying blockchain technology. This does not come as a surprise as the main bulk of work happens off-chain, as we shall see. For example, it could be adapted to Ethereum's smart contract logic.

Paper organisation. Section 2 gives some background needed. Section 3 introduces transaction chains and discuss their properties. Section 4 explains how to prove existence of transaction chains in Bitcoin using SNARKs. Finally, Section 5 presents our non-fungible token scheme NFT and analyses its security.

2 Preliminaries

2.1 Bitcoin Transactions

A transaction tx in the Bitcoin network has a unique identifier txid which is defined as the double $\mathsf{SHA256}$ of the serialized transaction data. The txid is not part of the transaction itself. A transaction input $\mathsf{tx.in}$ contains several fields. Including a reference to previous transaction ID $\mathsf{prevtxid}$, an output index vout, (this pair is known as the *outpoint*), and the unlocking script `scriptSig`. A transaction output $\mathsf{tx.out}$ contains the index vout of the output, a locking script `scriptPubKey`, and the amount of satoshis value locked.

Embedding data in transactions Data can be inserted into transactions at several positions. Data payloads do not play a role in script validation as they can be embedded between OP_PUSH and OP_DROP codes or positioned often an OP_RETURN opcode. For example, it is possible to have a pay-to-public-key script P2PK with embedded data. During the transaction validation, the unlocking script of the spending transaction tx, and the locking script of the parent transaction prevtx are combined:

$$\langle \texttt{scriptSig} \rangle \,||\, \texttt{scriptPubKey} := \langle \sigma\ \texttt{sighash} \rangle \,||\, [\mathsf{P2PK}\ \mathsf{pk}]\ \texttt{OP\_RETURN}\ \langle \mathsf{data} \rangle \quad (1)$$

The unlocking script contains the signature σ, and the locking script is the P2PK script (with public key pk) and the embedded data. Note that we are not writing out explicitly the set of opcodes comprising [P2PK pk]. The script is evaluated in reverse polish notation in the stack of the Bitcoin engine starting by pushing the data of scriptSig to the stack. Once the P2PK script has been executed successfully, which includes the signature check, the script becomes OP_RETURN $\langle$data$\rangle$ The script execution terminates when opcode OP_RETURN is reached. Hence the appended data is never pushed to the stack, but stored in the Bitcoin network as part of tx.

Choosing which inputs and outputs are signed The result of the P2PK script evaluation above is either true or false. It depends on the signature check, which involves verifying the signature σ using public key pk. The message that is verified against σ is essentially a portion of the spending transaction tx. The parts of tx that are signed is controlled with the flag sighash. For example, if set to sighash_all||anyonecanpay in the unlocking script of the i-th input $\mathsf{tx.in}_i$, the signature verification discards all other inputs but uses all outputs of tx. This allows different parties to add inputs to partially signed Bitcoin transactions (PSBT) while fixing the outputs (destination addresses). We will make use of different sighash flags in our NFT scheme of Section 5. See Table 1 for all possible flag values.

Flag value	**Meaning**
sighash_all	Sign all inputs and all outputs
sighash_none	Sign all inputs and no output
sighash_single	Sign all inputs and the output with the same index
sighash_all||anyonecanpay	Sign its own input and all outputs
sighash_none||anyonecanpay	Sign its own input and no output
sighash_single||anyonecanpay	Sign its own input and the output with the same index

Table 1. Values of sighash flag in Bitcoin. Taken from [6].

2.2 Recursive SNARKs – Proof Carrying Data

Proof-carrying data (PCD) [14] allows to prove correct execution of distributed computations run in mutually distrustful nodes. It is a generalization of incrementally verifiable computation (IVC) [30]. It allows to prove that a value z_n is the result of applying a function F iteratively n times

$$z_n = F(z_{n-1}), \ldots, z_1 = F(z_0) \tag{2}$$

In short, $z_n = F^n(z_0)$. In blockchains, PCDs have already found applications, most notably Mina [25] and others [21,9,13].

PCD scheme and properties One can see (2) as the transcript T of a computation between n nodes where the i-th nodes applies F to its incoming message z_{i-1} to produce an outgoing message z_i. If this is the case, the whole transcript is said to be *compliant* with F. Likewise, z_n is compliant with F, if it is the output of a transcript T compliant with F. A PCD scheme is a triplet of algorithms $\mathsf{PCD} = (\mathbb{G}, \mathbb{P}, \mathbb{V})$ with the following interface:

– $\mathbb{G}(1^\lambda, F) \to (\mathsf{pk}, \mathsf{vk})$. It takes as input a security parameter λ, and a function F seen as an arithmetic circuit C_F. It produces a pair of verification and proving keys.
– $\mathbb{P}(\mathsf{pk}, z_n, (z_{n-1}, \pi_{n-1})) \to \pi_n$. It takes as public input the outgoing message z_n, and as private input the incoming message z_{n-1} and a proof π_{n-1}. It outputs a proof π_n for the F-compliance of z_n. Namely, a proof for the existence of messages $z_0, \ldots, z_{n-1}$ as in (2).
– $\mathbb{V}(\mathsf{vk}, z_n, \pi_n) \to \{\mathsf{accept}, \mathsf{reject}\}$. It takes the public input z_n and a proof π_n. It either accepts or rejects the proof for the F-compliance of z_n.

We informally state the properties of a PCD scheme, and refer to [5,2,14] for formal definitions. The scheme is *complete* if $\mathbb{V}$ always accepts proofs π_n generated by the honest prover $\mathbb{P}$. It is *succinct* if the size of π_n is $\text{poly}(\lambda, |C_F|)$. In particular the proof does not grow at each iteration, it only depends on the complexity of the arithmetic circuit for F. The scheme is *knowledge sound* if, given access to the random coins of any cheating prover $\mathbb{P}^*$ that produces a pair (z_n, π_n) accepted by the verifier $\mathbb{V}$, it is possible to extract a compliant transcript T as in (2) with overwhelming probability in the security parameter λ.

Constructing PCDs from SNARKs PCDs can be constructed in two ways. It was first realized in [2] from preprocessing succinct non-interactive arguments of knowledge (SNARKs) [4] with sublinear verification time (also known as *succinct* verification) over a cycle of elliptic curves. Given a preprocessing $\mathsf{SNARK} = (\mathbf{G}, \mathbf{P}, \mathbf{V})$, the statement $z_n = F^n(z_0)$ is split in two parts. The first part proves existence of z_{n-1} such that $z_n = F(z_{n-1})$; this is done by expressing F as an arithmetic circuit C_F, that is satisfied on public input z_n and private input z_{n-1} only if $z_n = F(z_{n-1})$. The second part proves existence of a valid proof

π_{n-1} attesting to the correctness of the $n-1$ previous iterations. This is done expressing the verifier $\mathbf{V}$ as a circuit $C_{\mathbf{V}}$, satisfied on public input vk and private inputs z_{n-1}, π_{n-1}, only if $\mathbf{V}(\mathsf{vk}, z_{n-1}, \pi_{n-1}) = \mathsf{accept}$, where vk is the preprocessed verification key generated by $\mathbf{G}$. The PCD prover proves satisfiability of both circuits C_F and $C_{\mathbf{V}}$ on public input z_n and private inputs z_{n-1}, π_{n-1} using vk as part of its proving key. Another way of constructing PCDs, initiated in Halo [10], is using a SNARK without succinct verification, but with an accumulation scheme that allows to accumulate the proofs at each step, and postpone the verification of the proofs. The requirement is that the accumulator must be succinctly verifiable [10,8,11]. Recently, PCDs have been constructed from folding schemes, without using SNARKs at all, in Nova [22].

3 Transaction Chains

In this section, we introduce our notion of *transaction chains*. In the next section, we show how to prove the existence of a chain succinctly. The inputs and outputs of a Bitcoin transaction form ordered sets. We write $\mathsf{tx.in}_s$ and $\mathsf{tx.out}_r$ to respectively denote the s-th input and r-th output.

Definition 1 (Transaction Chain). *Let* T *be a set of transactions. The sequence* $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n} := (\mathsf{tx}_0, \mathsf{tx}_1, \ldots, \mathsf{tx}_{n-1}, \mathsf{tx}_n)$ *with* $\mathsf{tx}_i \in \mathsf{T}$*, is a* transaction chain of length n *if for each* $1 \leq i \leq n$ *some of the inputs of* tx_i *references an output of* tx_{i-1}.

As mentioned in Section 2.1, in Bitcoin, a transaction tx_i references a parent transaction tx_{i-1} in its inputs using the identifier txid_{i-1}. We can rely on the collision resistance of the hash function to ensure distinct identifiers for all Bitcoin transactions. This ensures that any chain $\vec{\mathsf{c}}_{\mathsf{tx}_1 \to \mathsf{tx}_n}$ has no repeated transactions. Also, note that transaction chain is a transitive relation, therefore we can concatenate two chains $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$, $\vec{\mathsf{c}}_{\mathsf{tx}_n \to \mathsf{tx}_m}$ to obtain a longer transaction chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_m}$ (i.e., by removing the first transaction of the second chain $\vec{\mathsf{c}}_{\mathsf{tx}_n \to \mathsf{tx}_m}$).

We note that there could be more than one chain linking two transactions. For example, consider the set $\mathsf{T} = \{\mathsf{tx}_0, \mathsf{tx}_1, \mathsf{tx}_2, \mathsf{tx}_3\}$, where tx_0 has two outputs, the first output is spent by tx_1, and the second output by tx_2. If tx_3 spends both tx_1 and tx_2, then we have two chains $\vec{\mathsf{c}}_1 = (\mathsf{tx}_0, \mathsf{tx}_1, \mathsf{tx}_3)$, and $\vec{\mathsf{c}}_2 = (\mathsf{tx}_0, \mathsf{tx}_2, \mathsf{tx}_3)$ linking tx_0 with tx_3.

We are also interested in chains that are unique. That is, given two transactions there exists only one possible way to link them. We address this issue fixing the inputs and outputs through which the link is established.

Definition 2 ((r,s)-Primary Chain). *Let a set of transactions* T *and a sequence* $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n} = (\mathsf{tx}_0, \ldots, \mathsf{tx}_n)$ *with* $\mathsf{tx}_i \in \mathsf{T}$*. We say* $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$ *is a* (r,s)-primary chain *if it is a transaction chain and the input* r*-th input of* tx_i *references the* s*-th output of* tx_{i-1}.

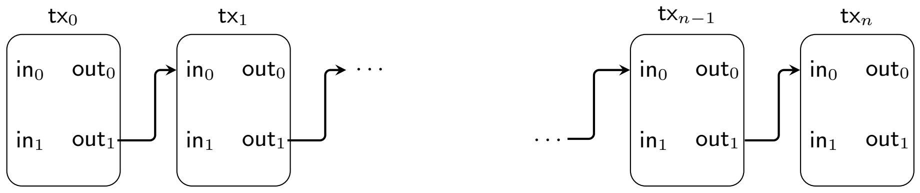

Fig. 1. A $(1,2)$-primary chain. The first input of tx_i spends the second output of tx_{i-1}.

In the lemma below we characterize the set of all possible (r,s)-primary chains starting at tx_0. Namely, there can only be chains that are subsequences of the largest chain in the set $\mathsf{C}^{(\mathsf{T}_\mathcal{B})}_{\mathsf{tx}_0\rightarrow}$, where:

$$\mathsf{C}^{(\mathsf{T}_\mathcal{B})}_{\mathsf{tx}_0\rightarrow} := \left\{ \vec{\mathsf{c}} = (\mathsf{tx}_0, \dots, \mathsf{tx}_n) \,\middle|\, \begin{array}{l} \vec{\mathsf{c}} \text{ is a } (r,s)\text{-primary chain (Defn. 2)} \\ \mathsf{tx}_i \in \mathsf{T}_\mathcal{B} \ \forall 0 \le i \le n \end{array} \right\} \tag{3}$$

Lemma 1 (Non-Diverging Primary Chains). *Let* $\mathsf{T}_\mathcal{B}$ *be the set of all transactions in the Bitcoin network. Let any transaction* tx_0 *in* $\mathsf{T}_\mathcal{B}$*, and let* $\vec{\mathsf{c}}^{\,max}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ *be an* (r,s)*-primary chain of maximal length. Then, the set* $\mathsf{C}^{(\mathsf{T}_\mathcal{B})}_{\mathsf{tx}_0\rightarrow}$ *of all* (r,s)*-primary transactions starting at* tx_0 *are subsequences of* $\vec{\mathsf{c}}^{\,max}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$*. That is:*

$$\mathsf{C}^{(\mathsf{T}_\mathcal{B})}_{\mathsf{tx}_0\rightarrow} = \{(\mathsf{tx}_0, \dots, \mathsf{tx}_i) \mid \mathsf{tx}_i \in \vec{\mathsf{c}}^{\,max}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}\}$$

Proof. This is a direct consequence of the double-spending resistance property of a blockchain. First observe that $\vec{\mathsf{c}}^{\,max}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ is of maximal length, so it cannot be a subsequence of any other primary chain. Now, if there exists an (r,s)-primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_m}$ that is not a proper subsequence of $\vec{\mathsf{c}}^{\,max}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$, then, since both chains share the same origin tx_0, at some point they must diverge. Say they are equal up to the i-th transaction tx_i. This means that the s-th output tx_i has been spent twice, which is not possible. □

The lemma above also proves that there is only one primary chain of maximal length. Not surprisingly, the largest (r,s)-primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ in Bitcoin is that whose end transaction tx_n has the s-th output unspent.

Lemma 2 (Largest Primary Chain). *Let* $\mathsf{T}_\mathcal{B}$ *be the set of all transactions in the Bitcoin network. Let any* tx_0 *in* $\mathsf{T}_\mathcal{B}$*, and let* $\vec{\mathsf{c}}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ *be an* (r,s)*-primary chain such that the s-th output of* tx_n *is unspent. Then,* $\vec{\mathsf{c}}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ *is the largest* (r,s)*-primary chain starting at* tx_0*.*

Proof. Lemma 1 shows that all (r,s)-primary chains must be subsequences of the largest (r,s)-primary chain. Now, $\vec{\mathsf{c}}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ cannot be a subsequence of any other primary chain because the s-th output of tx_n is unspent. We conclude that $\vec{\mathsf{c}}_{\mathsf{tx}_0\rightarrow\mathsf{tx}_n}$ is the largest chain starting at tx_0. □

4 Succinct Proofs for Transaction Chains

Consider the following scenario with a prover Alice and a verifier Bob. The first transaction tx_0 is already in the blockchain $\mathcal{B}$ and is known to both Alice and Bob. Alice submits a transaction tx_n to $\mathcal{B}$ and also sends it to Bob. Both parties now have two transactions $(\mathsf{tx}_0, \mathsf{tx}_n)$. Alice wants to prove to Bob the statement:

$$\text{“}\mathsf{tx}_n \textit{ is linked to } \mathsf{tx}_0 \textit{ through a primary chain } \vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}\text{”} \tag{4}$$

Alice generates a proof π_n that attest to the veracity of her statement. If Bob accepts π_n as valid, he has the triplet $(\mathsf{tx}_0, \mathsf{tx}_n, \pi_n)$. Now consider a third actor Charlie. Bob creates a new transaction tx_{n+1} such that $(\mathsf{tx}_n, \mathsf{tx}_{n+1})$ is a primary chain. He amends Alice's proof π_n to create a new proof π_{n+1} to also show that tx_{n+1} is linked to tx_0 through a primary transaction chain. He sends $(\mathsf{tx}_{n+1}, \pi_{n+1})$ to Charlie. The information flow can be visualised in Figure 2. The point we are trying to make is that proof generation is incremental; new parties can come in, augment an existing chain in $\mathcal{B}$, and prove correctness of the augmentation based on older proofs.

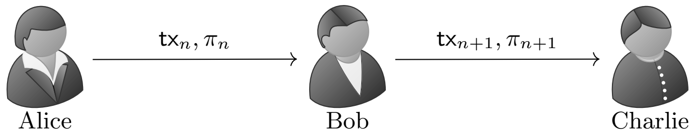

Fig. 2. Incremental chain augmentation. tx_0 is known to all parties. π_i proves existence of chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_i}$.

Trivial solution: Send the primary transaction chain. Alice simply sends the entire chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n} = (\mathsf{tx}_0, \dots, \mathsf{tx}_n)$ as her proof π_n. Bob can explicitly check that $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$ is a primary chain appearing in the blockchain $\mathcal{B}$. This solution requires communication and verification cost linear in the size of the chain $\mathcal{O}(n)$. It also incurs in repetition cost: Charlie has to check the entire chain again, even though Bob checked it already up to the n-th link.

Efficient solution: Use SNARKs. Alice sends to Bob the last transaction tx_n, and a short SNARK proof π_n attesting to the existence of $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$. Bob verifies π_n and checks that tx_n is in $\mathcal{B}$. The size of the transmitted proof is now $|\pi_n| = \mathcal{O}(\log(n))$, a significant improvement compared with the trivial solution above. See Table 2 for a comparison summary.

4.1 Proving Existence of Primary Chains Recursively

Recall from Section 3 that for a set of transactions $\mathsf{T}_{\mathcal{B}}$ in blockchain $\mathcal{B}$, and $\mathsf{tx}_0 \in \mathsf{T}_{\mathcal{B}}$, the set $\mathsf{C}^{(\mathsf{T}_{\mathcal{B}})}_{\mathsf{tx}_0 \to}$ denotes all the (r, s)-primary chains starting at tx_0.

Method	Communication	Verification
Send chain	$\mathcal{O}(n)$: send $n-1$ transactions	$\mathcal{O}(n)$: check $n-1$ transactions
SNARK-based	$\mathcal{O}(\log(n))$: send tx_n and π_n	$\mathcal{O}(\log(n))$: verify π_n

Table 2. Comparison of solutions to check primary chains $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$.

Thus, we can formalize the informal statement (4) in the following NP relation:

$$\mathcal{R}_{\mathsf{T}_{\mathcal{B}},\mathsf{tx}_0} := \left\{ (\mathsf{tx}_n; \vec{\mathsf{c}}) \;\middle|\; \begin{array}{l} \vec{\mathsf{c}} = (\mathsf{tx}_0, \mathsf{tx}_1, \ldots, \mathsf{tx}_{n-1}, \mathsf{tx}_n) \\ \vec{\mathsf{c}} \in \mathsf{C}^{(\mathsf{T}_{\mathcal{B}})}_{\mathsf{tx}_0 \to} \end{array} \right\} \tag{5}$$

In principle the prover needs the entire chain $\vec{\mathsf{c}}$ linking tx_n to tx_0 as private information to prove statement (4). He would somehow need to get the chain $\vec{\mathsf{c}}$ either from the previous prover or from $\mathcal{B}$. This poses a problem because the chain may be large and we would like to avoid sending it between parties. The observation is that *provided* the chain from tx_0 up to the parent tx_{n-1} is a primary chain, we just need to ensure the last link holds. We ensure the chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_{n-1}}$ is a primary chain validating a proof π_{n-1} for the parent transaction tx_{n-1} using a recursive SNARK.

Description of the circuit. The recursive circuit $\mathsf{C}_{\mathsf{ptc}}$ for primary transaction chains is described in Figure 3. Note that the circuit has hard-coded in its description the transaction identifier txid_0 of the first transaction tx_0 of the chain. Also, as described, the circuit uses a SNARK with succinct verification.

$\underline{\mathsf{C}_{\mathsf{ptc}}((\mathsf{tx}_n, b_{\mathsf{base}}); (\mathsf{tx}_{n-1}, \pi_{n-1}))\text{:}}$

Public input $\mathsf{tx}_n, b_{\mathsf{base}}$. The last transaction in the chain, and a 'base case' flag
Private input $\mathsf{tx}_{n-1}, \pi_{n-1}$. The parent transaction and (an optional) proof π_{n-1} for satisfiability of $\mathsf{C}_{\mathsf{ptc}}$ on public input tx_{n-1}.

```
1. Parse $\mathsf{tx}_n.\mathsf{in}_r = \mathsf{txid}||\mathsf{vout}$
2. Check $\mathsf{vout} = s$
3. If $b_{\mathsf{base}} = \mathsf{true}$ (base case), check $\mathsf{txid} = \mathsf{txid}_0$ // $\mathsf{txid}_0$ hard-coded
4. If $b_{\mathsf{base}} = \mathsf{false}$ (recursive case):
   (a) Check $\mathsf{txid} = \mathsf{SHA256d}(\mathsf{tx}_{n-1})$
   (b) Check $\pi_{n-1}$ is valid for public input $(\mathsf{tx}_{n-1}, b_{\mathsf{base}})$
```

Fig. 3. Circuit to prove existence of a (r, s)-primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$.

The recursive SNARK (PCD) for primary transaction chains Having described the circuit $\mathsf{C}_{\mathsf{ptc}}$, the preprocessing PCD to prove satisfiability of $\mathsf{C}_{\mathsf{ptc}}$ is the triplet of algorithms $\mathsf{PCD}_{\mathsf{ptc}} = (\mathbb{G}_{\mathsf{ptc}}, \mathbb{P}_{\mathsf{ptc}}, \mathbb{V}_{\mathsf{ptc}})$ with the following interface.

- $\mathbb{G}_{\mathsf{ptc}}(1^\lambda, \mathsf{txid}_0) \to (\mathsf{pk}, \mathsf{vk})$. On input a security parameter λ and the identifier of the first transaction txid_0 it outputs a pair of proving and verification keys pk, vk.
- $\mathbb{P}_{\mathsf{ptc}}(\mathsf{pk}, (\mathsf{tx}_n, b_{\mathsf{case}}); (\mathsf{tx}_{n-1}, \pi_{n-1})) \to \pi_n$. On input a proving key, the public input $(\mathsf{tx}_n, b_{\mathsf{case}})$, and the private input $(\mathsf{tx}_{n-1}, \pi_{n-1})$ it generates a proof π_n attesting to the existence of a chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$. Thus $(\mathsf{tx}_n, \vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}) \in \mathcal{R}_{\mathsf{T}_\mathcal{B}, \mathsf{tx}_0}$.
- $\mathbb{V}_{\mathsf{ptc}}(\mathsf{vk}, (\mathsf{tx}_n, b_{\mathsf{case}}), \pi_n) \to \{\mathsf{accept}, \mathsf{reject}\}$. On input a verification key vk and the public input $(\mathsf{tx}_n, b_{\mathsf{case}})$ it either accepts or rejects.

We emphasize that txid_0 needs to be known in advance to use it in step (3) of $\mathsf{C}_{\mathsf{ptc}}$. The later means that $\mathsf{PCD}_{\mathsf{ptc}}$ can be used only for the chain that starts at tx_0.

Theorem 1. *Let $\mathsf{T}_\mathcal{B}$ be the set of transactions in the Bitcoin network. If the verifier $\mathbb{V}_{\mathsf{ptc}}$ accepts a proof π_n for $\mathsf{tx}_n \in \mathsf{T}_\mathcal{B}$, then there exists a chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$ in $\mathsf{T}_\mathcal{B}$ with overwhelming probability.*

Proof. Using the soundness of $\mathsf{PCD}_{\mathsf{ptc}}$, it is enough to show that if the circuit $\mathsf{C}_{\mathsf{ptc}}$ accepts tx_n then it exists a chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$ with overwhelming probability. We will use the following claim that holds true in Bitcoin.

Claim. All coinbase transactions in the Bitcoin network have one unique outpoint whose parent ID reference is the zero-byte array.

The proof of the theorem is concluded proving the following lemma.

Lemma 3. *Let $\mathsf{T}_\mathcal{B}$ be the set of Bitcoin transactions at a given time. If $\mathsf{tx}_0 \in \mathsf{T}_\mathcal{B}$ is not an ancestor of $\mathsf{tx}_n \in \mathsf{T}_\mathcal{B}$, then $\mathsf{C}_{\mathsf{ptc}}$ rejects on public input tx_n. Thus, $\forall \mathsf{tx}_{n-1}, \pi, b_{\mathsf{base}}$ it holds $\mathsf{false} = \mathsf{C}_{\mathsf{ptc}}((\mathsf{tx}_n, b_{\mathsf{base}}), (\mathsf{tx}_{n-1}, \pi_{n-1}))$ with overwhelming probability.*

Proof. All transactions in Bitcoin originate from a coinbase transaction. Let $\vec{\mathsf{c}} = (\mathsf{tx}_{\mathsf{cb}}, \mathsf{tx}_1 \ldots, \mathsf{tx}_n)$ be an (r, s)-primary chain connecting a coinbase transaction $\mathsf{tx}_{\mathsf{cb}}$ with tx_n. Now, assume by contradiction that $\mathsf{C}_{\mathsf{ptc}}$ accepts tx_n. By hypothesis tx_0 is not an ancestor of tx_n, and we assume all transactions in $\mathsf{T}_\mathcal{B}$ have different identifiers. Therefore the base case of $\mathsf{C}_{\mathsf{ptc}}$ is not triggered, and there must be a valid proof π_{n-1} for the parent tx_{n-1}. Repeating this argument backwards, there must exist a valid proof $\pi_{\mathsf{cb}-1}$ for the parent of the coinbase transaction $\mathsf{tx}_{\mathsf{cb}}$. In other words, the prover used a 'parent' transaction $\mathsf{tx}_{\mathsf{cb}-1}$ for $\mathsf{tx}_{\mathsf{cb}}$ whose identifier is the zero-byte array — see the claim above. The later cannot occur in practice due to the collision resistance of $\mathsf{SHA256}$. This concludes the proof of the lemma and the theorem. □

4.2 Implementation Details and Benchmarks

Circuit logic. We have implemented several circuit gadgets to construct $\mathsf{C}_{\mathsf{ptc}}$ from Figure 3. For simplicity we set $r = s = 0$.

- $\mathsf{first\_outpoint\_OK}(\mathsf{tx}, \mathsf{txid})$. Evaluates to true iff $\mathsf{tx.in}_0 = \mathsf{txid}||0$

- $\mathsf{txid\_OK}(\mathsf{tx}, \mathsf{txid})$. Evaluates to true iff $\mathsf{txid} = \mathsf{SHA256d}(\mathsf{tx})$
- $\mathsf{proof\_OK}(\mathsf{tx}, \pi)$. Evaluates to true iff $\mathbf{V}(\mathsf{vk}, (\mathsf{tx}, \mathsf{false}), \pi) = \mathsf{accept}$. Here recall from Section 2.2 that $\mathbf{V}$ is the verification logic of the underlying SNARK used in the construction of the PCD, and vk the verification key.

Using the above gadgets, we can construct the following circuits for the base case and recursive case, respectively:

$$\mathsf{parent\_is\_txid_0}(\mathsf{tx}_n) := \mathsf{first\_outpoint\_OK}(\mathsf{tx}_n, \mathsf{txid}_0)$$
$$\mathsf{primary\_chain\_OK}(\mathsf{tx}_n, \mathsf{tx}_{n-1}, \pi_{n-1}) :=$$
$$\mathsf{txid\_OK}(\mathsf{tx}_{n-1}, \mathsf{txid}_{n-1}) \wedge \mathsf{first\_outpoint\_OK}(\mathsf{tx}_n, \mathsf{txid}_{n-1}) \wedge \mathsf{proof\_OK}(\mathsf{tx}_{n-1}, \pi_{n-1})$$

The circuit $\mathsf{C_{ptc}}$ is then set to:

$$\begin{aligned} &\mathsf{C_{ptc}}((\mathsf{tx}, b_{\mathsf{base}}); (\mathsf{tx}_{n-1}, \pi_{n-1})) := \\ &\quad (b_{\mathsf{base}} = \mathsf{true} \wedge \mathsf{parent\_is\_txid_0}(\mathsf{tx}_n)) \\ &\quad \vee \\ &\quad (b_{\mathsf{base}} = \mathsf{false} \wedge \mathsf{primary\_chain\_OK}(\mathsf{tx}_n, \mathsf{tx}_{n-1}, \pi_{n-1})) \end{aligned}$$

PCD scheme. We have implemented the SNARK scheme $\mathsf{PCD_{ptc}}$ defined in the previous section following the approach of [2]. Namely, using cycles of elliptic curves to implement the verification logic as part of the circuit in step (4b) of $\mathsf{C_{ptc}}$. The underlying SNARK is Groth16 [19]. Our implementation is written in Rust and uses arkworks library [1]. The cycle of curves is MNT4-MNT6. These are MNT curves [24] of embedding degrees 4 and 6, respectively. For production code, the size of the fields should be large, i.e. of size ≈ 750 bits.

In Table 3 we report benchmarks for the time it takes to prove and verify a proof. The tests have run in a laptop, 2019 MacBook Pro 2.6 GHz 6 Cores i7, 12 Threads, 32 GB Memory, and in an embedded processor (SoC) Raspberry Pi ARM BCM2835 1.80 GHz 1 Processor, 4 Cores, 4 Threads, 3.71 GB Memory. The most expensive gadgets are $\mathsf{txid\_OK}$ and $\mathsf{proof\_OK}$. The former needs to generate constraints for two evaluations of the hash function $\mathsf{SHA256}$, which is not zero-knowledge friendly. The later contains the logic of the Groth16 verifier $\mathbf{V}$. We have fixed the transaction size to 226 bytes – the minimum size of a Bitcoin transaction. This means the compression function of $\mathsf{SHA256}$ is applied twice to generate txid from tx.

5 An Application: NFTs with Atomic and Fully-Fair Swaps

We define a non-fungible token scheme (NFT) as a tuple of algorithms $\mathsf{NFT} = (\mathsf{mint}, \mathsf{list}, \mathsf{sell}, \mathsf{buy})$. In our scheme, minting a token tk essentially embeds tk in a transaction tx_0. A user with public key pk_U is the owner of tk, if there exist a (r, s)-primary transaction chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$, with the s-th output of tx_n unspent and controlled by pk_U. Owners can list their tokens for trading, and listed tokens can be exchanged placing sell or buy trade orders.

	Standard laptop (MacBook Pro)	**Smartphone (Raspberry Pi)**
Proving time	171 secs (≈ 3 mins)	970 secs (≈ 16 mins)
Verification time	3 secs	5 secs
Proof size	270 bytes	

Table 3. Times for proving and verifying satisfiability of circuit $\mathsf{C}_{\mathsf{ptc}}$ recursing Groth16 over MNT-753 cycle.

5.1 Description of the Scheme

Mint tokens. The issuer embeds the digital token tk in an *issuance* transaction tx_0. He then creates the SNARK proving key and verification key, and generates the *mint* transaction tx_1 as the first link of an (r, s)-primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_1}$ along with a proof π_1. Both transactions tx_0,tx_1 are uploaded to the blockchain and the mint process is finished. See Figure 4 for the algorithm mint.

Note that only the issuer can unlock the s-th output of tx_0 using his secret signing key sk_I. Hence, the issuer owns freshly minted tokens.

$\underline{\mathsf{mint}(1^\lambda, \mathsf{tk}, \mathsf{pk}_I, \mathsf{sk}_I)}$:

1. Create the **issuance transaction** tx_0 with token tk embedded in some of the outputs. (See equation (1) for `OP_RETURN` data). The s-th output locks no funds with a `P2PK` script using a public key pk_I controlled by the issuer.
2. Run the setup of the NFT program using the identifier txid_0 of tx_0:
$$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathbb{G}_{\mathsf{ptc}}(1^\lambda, \mathsf{txid}_0)$$
3. Create the **mint transaction** tx_1 whose r-th input spends the s-th output of tx_0. Unlock the r-th input using the signing key sk_I.
4. Create a proof for the (r, s)-primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_1} = (\mathsf{tx}_0, \mathsf{tx}_1)$, running:
$$\pi_1 \leftarrow \mathbb{P}_{\mathsf{ptc}}(\mathsf{pk}, (\mathsf{tx}_1, b_{\mathsf{case}}); (\mathsf{tx}_0, \pi_0)),$$
the boolean flag is set to $b_{\mathsf{case}} = \mathsf{true}$, and since this is the first link of the chain, the previous proof is empty $\pi_0 = \emptyset$.
5. Upload both transactions $\mathsf{tx}_0, \mathsf{tx}_1$ to the blockchain, and publish pk, vk.

Fig. 4. Minting tokens in Bitcoin.

List tokens (with off-chain receipts). Before a token tk can be traded, the keys pk, vk, and the first proof π_1 generated by the issuer are publicly announced. This can be done via embedding the keys in the mint transaction tx_1. Additionally, the data can stored off-chain such as a list maintained by a marketplace. We emphasize that the keys and the mint transaction tx_1 can be also published by the issuer. Ultimately, the issuer is responsible for authenticating this data, not the marketplace.

To mark a token ready for trading, the marketplace receives the transaction tx_n from the current owner (if $n = 1$, from the issuer), checks it is linked with the previous transaction tx_{n-1}, and if so generates the proof π_n (the off-chain receipt). The algorithm for listing is given in Figure 5.

```
$\underline{\mathsf{list}(\mathsf{L_{nft}}, \mathsf{tk}, n, \mathsf{tx}_n)}$:

1. If $n = 1$, add entry $(\mathsf{tk}, \mathsf{pk}, \mathsf{vk}, \mathsf{tx}_1, \pi_1)$ to list $\mathsf{L_{nft}}$. // $\mathsf{tx}_1$ is the mint transaction
2. Else $(n > 1)$ do:
   (a) Retrieve entry $\mathsf{e_{tk}} = (\mathsf{tk}, \mathsf{pk}, \mathsf{vk}, \mathsf{tx}_{n-1}, \pi_{n-1})$ from $\mathsf{L_{nft}}$
   (b) Check $r$-th input of $\mathsf{tx}_n$ spends $s$-th output of $\mathsf{tx}_{n-1}$. If not, abort and halt.
   (c) $\pi_n \leftarrow \mathbb{P}_{\mathsf{ptc}}(\mathsf{pk}, (\mathsf{tx}_n, \mathsf{false}); (\mathsf{tx}_0, \pi_{n-1}))$ // generate proof (off-chain receipt)
   (d) Update entry $\mathsf{e_{tk}} = (\mathsf{tk}, \mathsf{pk}, \mathsf{vk}, \mathsf{tx}_n, \pi_n)$ in $\mathsf{L_{nft}}$
```

Fig. 5. Listing tokens in an marketplace with off-chain receipts.

Transfer tokens. We explain how to transfer tokens via an (r, s)-primary chain with $r \neq s$. For simplicity we use a $(1, 2)$-primary chain as in Figure 1. The trade is a non-interactive process that is fair for both parties, the seller and the buyer. The outcome of the process is a single transaction tx_{n+1} with two inputs and two outputs. The first output pays the seller, and the second output transfers token ownership to the buyer. If tx_{n+1} is accepted in the blockchain, the seller gets paid and the buyer is the new owner of the token (via the $(1, 2)$-primary chain). Otherwise, none of them gets anything.

We detail two flavours of transfers (trading orders). In sell orders, the seller initiates the trade and sets the offer price sats. In buy orders, the buyer initiates, and sets the bid price. In either case, the initiator of the trade creates a partially-signed bitcoin transaction (PSBT) tx_{n+1}, and sends it to the other party, who finalizes it filling the remaining inputs and outputs. The inputs of the transaction are unlocked signing with an appropriate combination of `sighash` flags for security. The exchanged information flows in one direction (one round), which means the parties do not need to be online at the same time. See Figure 6 for the algorithms sell, buy.

5.2 Fairness for the Buyer and Seller

We deem an NFT scheme *correct* if there cannot be multiple legitimate owners of the same token tk at a given time. We say the exchange is *fair* if, provided a sell or buy trade transaction tx_{n+1} is accepted in the blockchain, then the buyer is the new owner of the token tk, and the seller is rewarded in sats.

$\underline{\mathsf{sell}(\mathsf{sats}, \mathsf{pk}_S, \mathsf{pk}_B, \mathsf{tx}_n, \pi_n, \mathsf{vk}):}$

– Seller inputs: $\mathsf{sats}, \mathsf{pk}_S, \mathsf{tx}_n$
– Buyer inputs: $\mathsf{pk}_B, \mathsf{tx}_n, \pi_n, \mathsf{vk}$

Offer: The seller initiates the trade.

1. Create a PSBT tx_{n+1} with the first input spending the second output of tx_n. The first output of tx_{n+1} locks sats (the offer price) with his public key pk_S.
2. Unlock the first input of tx_{n+1} signing with `sighash_single||anyonecanpay`. From now on, the first output of tx_{n+1} cannot be changed, but more inputs and outputs can be added.
3. Publicly announce the PSBT tx_{n+1}.

Finalize: The buyer settles the trade.

1. Check $\mathsf{accept} \overset{?}{=} \mathbb{V}_{\mathsf{ptc}}(\mathsf{vk}, (\mathsf{tx}_n, \mathsf{false}), \pi_n)$. If the proof is not valid, reject and halt.
2. Add a a second input to tx_{n+1} with funds sats, and a second output to tx_{n+1} locked with his public key pk_B.
3. The second input is unlocked signing with `sighash_all` or with `sighash_single`. From now on, no input, in particular the first one, nor the second output can be changed.
4. The transaction tx_{n+1} is ready

$\underline{\mathsf{buy}(\mathsf{sats}, \mathsf{pk}_S, \mathsf{pk}_B, \mathsf{tx}_n, \pi_n, \mathsf{vk}):}$

– Seller inputs: $\mathsf{pk}_S, \mathsf{tx}_n$
– Buyer inputs: $\mathsf{sats}, \mathsf{pk}_B, \mathsf{tx}_n, \pi_n, \mathsf{vk}$

Bid: The buyer initiates the trade.

1. Check $\mathsf{accept} \overset{?}{=} \mathbb{V}_{\mathsf{ptc}}(\mathsf{vk}, (\mathsf{tx}_n, \mathsf{false}), \pi_n)$. If the proof is not valid, reject and halt.
2. Create a PSBT tx_{n+1} with the first input spending the second output of tx_n. The first output of tx_{n+1} is left unspecified.
3. Add a second input funding tx_{n+1} with sats (the bid price). Add a second output locked with his public key pk_B.
4. Unlock the second input signing with `sighash_single`. From this point on, neither the second output nor the two inputs can be changed, but more outputs can be added.
5. Publicly announce the PSBT tx_{n+1}.

Finalize: The seller settles the trade.

1. Add the first output of tx_{n+1} locking sats with his public key pk_S.
2. Unlocks the first input of tx_{n+1} signing with `sighash_all` (or any other flag that signs its own output).
3. The transaction tx_{n+1} is ready

Fig. 6. Selling and buying tokens without control delegation in Bitcoin.

Correctness. The correctness of our scheme $\mathsf{NFT} = (\mathsf{mint}, \mathsf{list}, \mathsf{sell}, \mathsf{buy})$ is due to the non-diverging property of primary chains (cf. lemma 1). Primary chains originating at a given transaction tx_0 can only be subsequences of the largest primary chain. Put differently, there cannot be 'forked' chains.

Fairness for the buyer. This is achieved due to two observations. First, since the buyer successfully verifies the proof π_n of the seller, then there exists a primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_n}$; this follows from the soundness of $\mathsf{PCD}_{\mathsf{ptc}}$ and theorem 1. Therefore, there also exists a $(1, 2)$-primary chain $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_{n+1}}$ because the buyer explicitly adds the second output of tx_{n+1} (in both, sell and buy orders). Second, at the time tx_{n+1} is accepted in the blockchain, since its first output is unspent, it is guaranteed that $\vec{\mathsf{c}}_{\mathsf{tx}_0 \to \mathsf{tx}_{n+1}}$ is the largest primary chain (cf. lemma 2); note the second output of tx_{n+1} is controlled by the public key pk_B of the buyer, and the first input is always signed by the buyer when he funds tx_n (signing with either `sighash_all` or `sighash_single`). If the seller sends the token to someone else in between, the first input of tx_{n+1} is a double spend, so it will not be accepted in the blockchain. If someone else changes the second output (so the buyer would not acquire ownership), his payment will not go through neither.

Fairness for the seller. This trivially holds because the seller always locks the payment sats in the first output of tx_{n+1}, which is always signed (using either `sighash_single`||`anyonecanpay` in sell orders, or any other flag that signs the first output in buy orders); if the buyer does not fund tx_{n+1} properly, it will never be accepted on-chain, and the seller still owns the token because the second output of tx_n remains unspent.

5.3 Further Remarks

Identifying corrupted users. The seller can prove knowledge of the signing key needed to unlock the first input of the last traded transaction tx_n to the NFTM when he engages in the list algorithm (e.g., by signing a challenge message). The buyer can also prove the knowledge of the signing keys that unlocks the sats funding the trade transaction tx_{n+1} when she initiates a buy order or finalises a sell order.

Trade latency. The main overhead of our NFT scheme is when listing tokens. Therein, the off-chain proof receipt π_n for the previous transaction trade tx_n is generated. We report roughly three minutes runtime for the prover $\mathbb{P}_{\mathsf{ptc}}$ to generate π_n for the previous trade transaction tx_n (see Table 3). However, since π_n is *independent* of tx_{n+1}, the algorithm list can be executed well ahead of time. Due to the full fairness of our NFT scheme, the buyer only needs to check π_n in a sell or buy order. This means that from users perspective, the trade is done almost instantaneous, which is the time to verify the proof π_n with $\mathbb{V}_{\mathsf{ptc}}$.

Royalties and wash trading Each transaction in a chain can have a distinguished output with a specific amount locked by a fixed public key pk_I controlled by the NFT issuer. Similarly, wash trading can be mitigated by putting a cap in the number of trades for which a proof receipt can be generated. Both features can be encoded in the primary chain circuit $\mathsf{C}_{\mathsf{ptc}}$ of Figure 3. If the fee is not paid, or the trade counter reaches the upper bound, users will not be able to generate the off-chain proof receipt.

References

1. `Arkworks` zksnark ecosystem. `https://arkworks.rs` (2023)
2. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: CRYPTO Conference. Springer (2014)

3. Besançon, L., Da Silva, C.F., Ghodous, P., Gelas, J.P.: A blockchain ontology for dapps development. IEEE Access (2022)
4. Bitansky, N., Canetti, R., Chiesa, A., Goldwasser, S., Lin, H., Rubinstein, A., Tromer, E.: The hunting of the SNARK. IACR Cryptol. ePrint Arch. (2014)
5. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: STOC. ACM (2013)
6. Bitcoin SV Wiki. `https://wiki.bitcoinsv.io/index.php/SIGHASH_flags`
7. Blancaflor, E., Aladin, K.: Analysis of the nft's potential impact in an e-commerce platform: A systematic review. In: Proceedings of the 10th International Conference on Computer and Communications Management. ACM (2022)
8. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Halo infinite: Proof-carrying data from additive polynomial commitments. In: CRYPTO 2021. Springer (2021)
9. Bonneau, J., Meckler, I., Rao, V., Shapiro, E.: Coda: Decentralized cryptocurrency at scale. IACR Cryptology ePrint Archive (2020)
10. Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive proof composition without a trusted setup. IACR Cryptology ePrint Archive (2019)
11. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Proof-carrying data from accumulation schemes. IACR Cryptol. ePrint Arch. (2020)
12. Chaparala, H.K., Doddala, S.V., Showail, A., Singh, A., Gazzaz, S., Nawab, F.: Liftchain: A scalable multi-stage nft transaction protocol. In: 2022 IEEE International Conference on Blockchain (Blockchain) (2022)
13. Chen, W., Chiesa, A., Dauterman, E., Ward, N.P.: Reducing participation costs via incremental verification for ledger systems. IACR Cryptology ePrint Archive (2020)
14. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: Innovations in Computer Science - ICS. Proceedings. Tsinghua University Press (2010)
15. Das, D., Bose, P., Ruaro, N., Kruegel, C., Vigna, G.: Understanding security issues in the NFT ecosystem. CoRR (2021)
16. Entriken, W., Shirley, D., Evans, J., Sachs, N.: ERC-721: Non-fungible token standard. EIP (2018), `https://eips.ethereum.org/EIPS/eip-721`
17. Ordinal inscription. `https://ordinals.com/` (2023)
18. Fowler, A., Pirker, J.: Tokenfication - the potential of non-fungible tokens (nft) for game development. In: Annual Symposium on Computer-Human Interaction in Play. ACM (2021)
19. Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology - EUROCRYPT 2016. Springer (2016)
20. Guo, H., Chen, M., Ou, W.: A lightweight nft auction protocol for cross-chain environment. In: Machine Learning for Cyber Security. Springer Nature Switzerland (2023)
21. Kattis, A., Bonneau, J.: Proof of necessary work: Succinct state verification with fairness guarantees. IACR Cryptology ePrint Archive (2020)
22. Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. In: CRYPTO 2022. Springer Nature Switzerland (2022)
23. Marlinspike, M.: My first impressions of web3. `https://moxie.org/2022/01/07/web3-first-impressions.html` (2022)
24. Miyaji, A., Nakabayashi, M., Nonmembers, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (2001)
25. O(1) Labs: Mina cryptocurrency. `https://minaprotocol.com` (2017)

26. Ordinal theory handobbok. https://docs.ordinals.com/ (2023)
27. Park, A., Kietzmann, J., Pitt, L., Dabirian, A.: The evolution of nonfungible tokens: Complexity and novelty of nft use-cases. IT Professional (2022)
28. Park, S., Lee, J.H., Lee, S., Chun, J.H., Cho, H., Kim, M., Cho, H.K., Moon, S.M.: Beyond the blockchain address: Zero-knowledge address abstraction. Cryptology ePrint Archive (2023)
29. Radomski, W., Cooke, A., Castonguay, P., Therien, J., Binet, E., Sandford, R.: ERC-1155: Multi token standard. EIP (2018), https://eips.ethereum.org/EIPS/eip-1155
30. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Fifth Theory of Cryptography Conference, TCC. Springer (2008)
31. Vasan, K., Janosov, M., Barabási, A.L.: Quantifying NFT-driven networks in crypto art. Scientific Reports (2022)
32. Wang, Q., Li, R., Wang, Q., Chen, S.: Non-fungible token (NFT): overview, evaluation, opportunities and challenges. CoRR (2021)
33. Wu, B., Wu, B.: NFT: Crypto As Collectibles. Apress (2023)