Multi-signature NFT minting system on Cardano

Cardano Solutions.

2022/07/02

Abstract

Cardano is an open-source UTXO model based blockchain and smart contract platform, which internal cryptocurrency is ADA(₳). Contrarily to others, as Ethereum, it natively features minting, burning and sending tokens without implementing a contract [1]. Metadata can be included when minting a token which enables the creation of NFT (non fongible token). Conventionally on Cardano, NFT collections minting process is for a user to send a quantity of ADA to a specific address, and for the creator to distribute assets accordingly. This process is problematic for several reasons. First, it is highly centralised and requires trust involved as the creator can decide to not send the NFT, or the wrong one. Second, it is inefficient in terms of fees, time and network congestion, as back-and-forth it generates twice as much transactions. Also Cardano minimum UTXO amount requirements forces the creator to custody a certain quantity of user's ADA before sending the token. A single multi-signed minting transaction is an efficient workaround that is presented in this article.

1 Introduction

1.1 Cardano native assets

A Cardano wallet is a set of private and public keys generated using Ed25519 cryptographic elliptic curve from a seed (in practice, mnemonic words) [2], as described in BIP32-Ed25519 [3]. Unlimited amount of addresses can then be derived from those root keys, following CIP-1852 [5]. A creator who wants to emit an NFT collection first need to create such a wallet.

Every Cardano native asset is minted under a policy. A policy is a script, a set of rules, such as: a limitation to when assets can be minted, an upper bound for the number of assets that can be minted, and, most importantly, the hash of policy owner's public key. Only the owner of the private key associated with that public key can mint or burn token under this policy. Any NFT collection can be identified [6] by the BLAKE2 hash [7] of this script, more commonly named Policy Id. Therefore, asset supply management on Cardano is, by construction, a somewhat centralised process as it requires the policy owner to sign any minting or burning transaction.

1.2 Conventional NFT collection minting approach

The traditional approach to NFT collection minting on Cardano is a two transactions process. Creator generates a wallet ($Key^{pub}_{creator}$, $Key^{prv}_{creator}$), a policy (Pol_{id} containing $Key^{pub}_{creator}$) and two addresses ($Addr_{tresury}$, $Addr_{mint}$). User also creates a wallet (Key^{pub}_{user}, Key^{prv}_{user}) and an address $Addr_{user}$ filled with ADA.

User creates a transaction Trx_1 sending ADAs from $Addr_{user}$ to $Addr_{mint}$ and signs it using Key^{prv}_{user} before submitting it a node. Once creator observes that Trx_1 has been included into the ledger, he creates a second transaction Trx_2: minting the corresponding amount of NFTs under policy Pol_{id}, plus minimum UTXO ADA amount (see [1]) from $Addr_{mint}$ to $Addr_{user}$, and ADA profit from $Addr_{mint}$ to $Addr_{treasury}$. He then signs it with $Key^{prv}_{creator}$, which enables both ADA owned on $Addr_{mint}$ to be spent and NFT to be rightfully minted under the policy Pol_{id}. Trx_2 is then submitted to the network. If no more NFT are left to be minted creator refunds the user minus the back-and-forth fees.

Figure 1: Traditional NFT minting user interface.

1.3 Underlying issues

This minting process is highly inefficient not only for the user but also for the creator and the network itself. The main problem of this system is their is no trivial way to implement a choice of the assets the user will receive, this is why most of NFT drops based on it are random, but this is not the only issue.

Obviously, as two chained transactions are needed, transaction fees must be paid back-and-forth, and process takes twice the time as one transaction. The blockchain network also suffers from this as it gets very congested during NFT mints, which has been a serious concern for Cardano Network health in the past months, as mempool saturation is consistently historically high.

Also, an unreasonable amount of trust from the user into the creator is required as there is no guaranty that the creator will send back the good assets, or any asset at all after receiving payment. Moreover, if by error a bad amount of ADA is sent by user for payment or if no more NFT are left to be minted, there is a need for a refund transaction. Fees are once again paid twice for nothing, and network gets congested.

Another problem is that as Cardano requires a minimum ADA requirement per UTXO [1], creator at some point has custody of ADAs belonging to the user as they are needed to be locked with assets when sent back. This can be a problem with some country's legislation. For instance, in France, you have to have the PSAN (prestataire de services sur actifs numériques) status to do custody.

A more anecdotal issue resides in the fact that such a user experience is very different from a smooth wallet extention dApp connector experience seen on other chains, with for instance NFT minting on Metamask for Ethereum, or Phantom for Solana.

2 Mutli-signature minting transaction based approach

All the problems described above can be solved using a mutli-signature minting transaction.

2.1 Cardano transactions and the eUTXO model

As Bitcoin does, Cardano relies on the UTXO (unspent transaction output) model instead of Ethereum's account model to solve the double spending problem. Every transaction on the ledger consumes outputs from previous transactions, and generates new outputs called UTXOs.

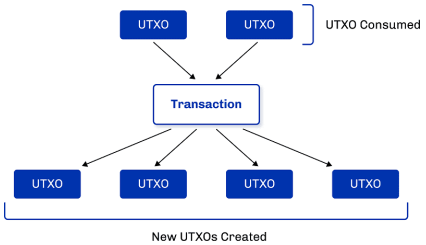

Figure 2: UTXO model diagram. Source: [8]

An UTXO is fully charcterised by: the address that owns it, the BLAKE2 hash of the transaction that created it, its index as an output of this transaction. More specifically, Cardano is based on the extended UTXO model [9], an incremented version with two major differences:

- The eUTXO model enables outputs to contain any metadata, instead of just address and coin value.
- More relevantly, not only can an address (an UTXO owner), be a public key, with the associated private key allowing owned UTXO consumption, but more generally an address can be a script which output allows or not the owned UTXO consumption. Concretely the authorisation to consume an UTXO as an input is called a witness ([6] p.26), and a valid signed transaction is composed of a transaction body provided with witnesses for each of its input UTXOs.

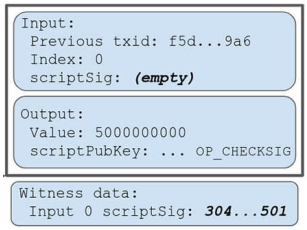

Figure 3: Valid signed transaction. Source: [10]

2.2 Multi-signature minting transaction

A transaction can contain as many witnesses as needed for every address, whether a script or not, involved. A multi-signature transaction is a transaction that is provided with multiple witnesses. A minting (resp. burning) transaction is a transaction, provided with a policy script, where a native asset is present in one (resp. none) of the output UXTOs' extended metadata whereas it was not present in any (resp. one) of the inputs. For it to be signed and valid it needs to be provided with said policy script witness, as explained above.

A mutli-signature minting transaction is simply a combinaison of both concepts. Such an example is represented in Figure 4, with same notation as defined in section 1.2:

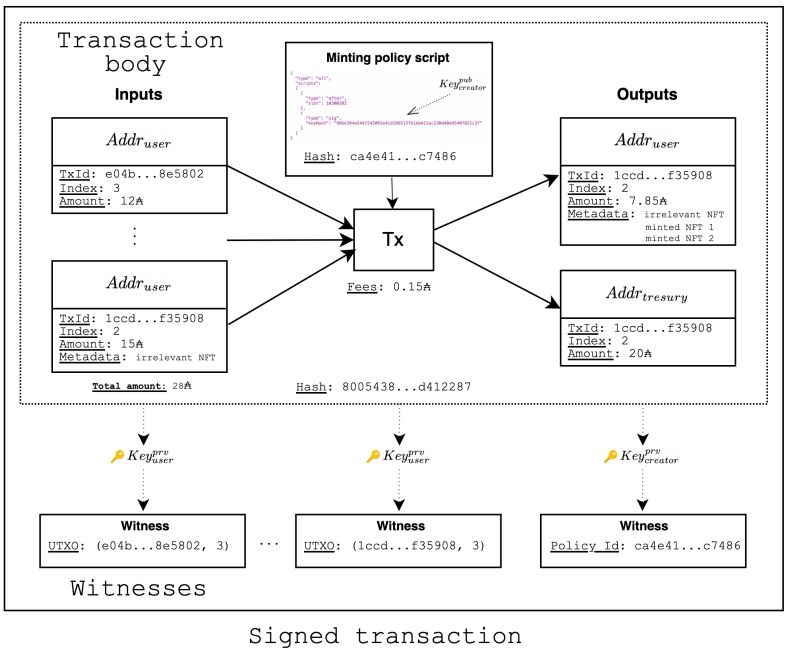

Figure 4: Diagram representing the multi-signed minting transaction with id 8005438...d412287.

2.3 Considered minting process

The approach examined here is to have a single transaction for minting the tokens and transferring coins from the user to the creator. The process is fully detailed in Figure 5, and the transaction in Figure 4.

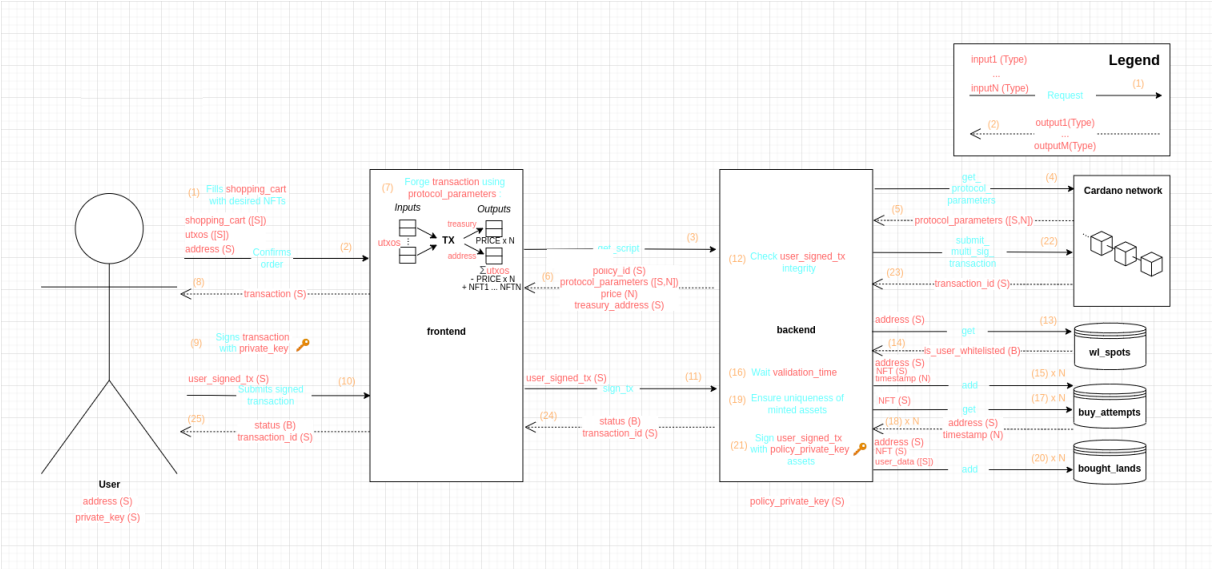

Figure 5: Multi-signature minting process diagram

Preparing parameters: User selects any amount of assets he desires to receive by filling a shopping cart (step 1). When cart is confirmed (step 2), protocol parameters (steps 4, 5), policy script and chosen NFT prices, which will determine payment amount, are fetch from backend (steps 3, 6).

Forging and user signing: Transaction inputs are then be chosen among user's address owned UTXOs using a coin selection algorithm (such as Random Improve, see CIP-2 [5]) to at least match payment amount. Two outputs are included in transaction. One with user change and minted NFTs, one with payment to treasury address. Transaction body is provided with policy script and metadata corresponding to minted assets. Fees are calculated using protocol parameters. Subsequently, transaction is forged (steps 7), signed with Key^{prv}_{user} by user (step 8, 9), and sent to backend (steps 10, 11).

Creator signing: It is then sent back to creator's backend where its integrity is checked (step 12), and signed with $Key^{prv}_{creator}$ (step 21), after checking whitelist (steps 13, 14), and ensuring NFT uniqueness (steps 15 to 20).

Transaction subimission: Transaction is then broadcasted to the Cardano network through a node run in the backend (steps 22, 23). Status (success or fail) and transaction hash can then be returned to the user (steps 24, 25).

2.4 User experience

On the user side, the experience is simply to click on a "Mint" button and sign a transaction using a dApp connector wallet extension such as Nami, Eternal or Yoroi as shown in Figure 6. He can check transaction metadata and be sure that he will receive the NFT he asked for. If the transaction does not reach the ledger, he won't lose any ADA, neither payment nor fees. It is not possible for him to send the wrong amount or mint if there are no NFT left as creator's backend would refuse to sign the transaction.

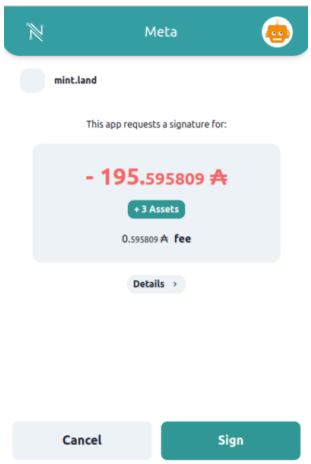

Figure 6: Multi-signature minting transaction user interface on Nami wallet

3 Improvement proposals

3.1 Whitelist token

Whitelist enables the creator to restrict users access to mint and grant some user a preferential price. In the considered protocol, the whitelist is handled in creator's backend which is not transparent to users, and hard to handle for creator as it must ensure every user registered correctly to the whitelist, with a compatible wallet.

A solution would be to generate a fongible token that acts as a whitelist spot and that would be burned in the transaction, and send it to every whitelisted user. The whitelist spot could then be exchanged between users addresses(or not if the creator decides sending the whitelist token invalidates it), but most importantly, everybody could monitor the supply and distribution of whitelist spots.

3.2 Reveal system

By construction, this multi-signature minting system does not allow for a random/unknown NFT to be minted by a user as he will always be able to see the metadata of the transaction before signing it. He could the just ask for new transaction until getting a specific asset.

This issue can be solved by splitting the mint into two transaction. A first transaction allows user to mint a temporary NFT containing a hexadecimal hash as a metadata under the wanted final policy. A second "reveal" multi-signature minting and burning transaction would enable the user to burn the intermediate NFT and mint the definitive one. The hexadecimal hash contained in the burnt NFT would be the BLAKE2 hash of the metadata of the definitive NFT, as the mapping would have been made previously to the first mint by the creator.

3.3 Creator signature using a smart contract

Though no trust is required for the user, as the max supply, and minting time limit can be enforced in the policy script, a major issue with proposed protocol still lies in the degree of centralisation of the minting process as it requires a signature by creator's policy owning wallet. A huge improvement would be to write a Cardano smart contract handling the policy witnessing, instead of the centralised creator backend.

Remark: There is an incompatiblity between section 3.2 improvement proposal and the section 3.3 one as a Cardano smart contract is deterministic. Calling an RNG (random number generator) oracle would solve this issue but would bring back a centralised actor in the process, unless the RNG oracle is a decentralised protocol itself.

References

[1] Andre Knispel, Polina Vinogradova
A Formal Specification of the Cardano Ledger with a Native Multi-Asset Implementation (2020-07-31),
https://hydra.iohk.io/build/5949624/download/1/shelley-ma.pdf

[2] Adrestia team, IOHK
Cardano Wallet documentation - HD Wallets (2020-10-21),
https://input-output-hk.github.io/adrestia/cardano-wallet/concepts/hierarchical-deterministic-wallets

[3] Dmitry Khovratovich: University of Luxembourg, Jason Law: Evernym, Inc.
Hierarchical Deterministic Keys over a Non-linear Keyspace,
https://input-output-hk.github.io/adrestia/static/Ed25519_BIP.pdf

[4] Adrestia team, IOHK
Cardano Wallet documentation - Address Derivation (2020-10-21),
https://input-output-hk.github.io/adrestia/cardano-wallet/concepts/address-derivation

[5] Matthias Benkort, Duncan Coutts, Sebastien Guillemot, Sebastien Guillemot, Frederic Johnson, Robert Phair
Cardano Improvement Proposal,
https://cips.cardano.org/

[6] Jared Corduan, Matthias Gudemann, Polina Vinogradova
A Formal Specification of the Cardano Ledger (2019-10-15),
https://hydra.iohk.io/build/4975913/download/1/ledger-spec.pdf

[7] M-J. Saarinen, Ed.: Queen's University Belfast, J-P. Aumasson: Kudelski Security
The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC) (2015-11),
https://datatracker.ietf.org/doc/html/rfc7693

[8] IOHK
Cardano Plutus documentation (2022-03-02),
https://docs.cardano.org/plutus/eutxo-explainer

[9] Manuel M.T. Chakravarty, James Chapman1, Kenneth MacKenzie, Michael Peyton Jones, and Philip Wadler: IOHK. Orestis Melkonian: University of Edinburgh
The Extended UTXO Model (2019-10-15),
https://api.zotero.org/groups/478201/items/T24L95MI/file/view?key=Qcjdk4erSuUZ8jvAah59Asef

[10] Daniel Won
What is SegWit? SegWit Explained in a Way You Can Understand (2020-05-15),
https://www.exodus.com/news/segwit-explained/