Adding randomness to the NFT Swap Infrastructure templates project

NFT Swap infrastructure templates project at a glance

In the first phase of the project, we created a total solution for enabling NFT projects to create so-called swap pools. These let the members of the NFT project community swap specific NFTs that belong to the same policy id, meaning they were minted with the same script.
We provided this to the community in an open source Github repository in a format design for easy reuse and configuration. We called these components of phase 1; **the specific swap templates**. The solution consists of a dApp that contains an admin module and a swap interface. The Admin module lets you build and configure the swap pool using one of our three pool smart contract templates and to add and remove NFTs to these smart contracts. The pool smart contracts are built for you directly and are tightly linked to the wallet you have connected to the dApp. The swap UI lets you explore the Swap pool NFTs and swap your NFTs for another as long as they belong to the same policy.
We developed three smart contracts to allow for full flexibility:
SpecificSwap which allows transactions that take X NFTs from the pool as long as also X NFTs are sent to the contract. Additionally, X must be more than 0 and the transaction only moves NFTs of the defined policy id in or out of the contract.
SpecificSwapFiltered which is a variant of SpecificSwap that in addition only allows the swap transaction if the NFTs moved have a name contained in the list of allowed NFT names to swap.
SpecificSwapTokenNameRule is also a variant of SpecificSwap, but it is set up with name and number ranges for the NFTs so for example it can be said that the smart contract allows swap if it has a correct name and number is between 1 and 100, without needing to list all 100 NFTs, only start number 1 and end number 100.
All code from phase 1 can be explored at
https://github.com/NFT-Guild/NFT-Swap-Infrastructure-Templates/tree/main

Time to add randomness to the mix

With phase 2 of this initiative, we want to extend the project templates with random swaps, meaning that the user should be able to select an NFT they want to send to the swap pool, but they will not know which NFT they will get in return as is the case in our existing specific swap templates.

Technically and theoretically, randomness is not clearly defined. The level of predictability extends from pseudo randomness where randomness is good enough for some low risk use cases but unusable for high risk use cases where large sums of money is involved. Pseudo randomness could mean that you receive a seemingly random result when asking for an item or number from a collection, but if you for example ask again at the same time the day after, you will receive the same item or number returned. Or said differently, the random function follows a pattern that can be understood and therefore be exploited. In our project we will describe different approaches that add randomness to our swap pools and discuss Pros and Cons for each approach.

Randomness on Cardano

By design, the EUTxO model of Cardano is deterministic and smart contracts therefore cannot behave in an indeterministic manner. Validation of transactions is determined by the rules of the smart contract script and these rules are static and unchangeable once they have been built. To achieve randomness despite this static nature, we need to give the smart contract rules access to a randomized component when doing the validation of a transaction. The random component must therefore be provided to the script as a parameter that will change from one usage to the next. Because of security considerations, smart contracts on Cardano are also not allowed to access off-chain resources, so this random data must be on-chain. Cardano lets us provide data to the smart contract using so-called reference UTxOs that can contain additional data.

Pseudo-randomness and true randomness

To do fair, random swaps, we need to provide a random component to the swap operation so it is infeasible for all involved parties to predict which NFT is returned from the swap pool on each swap interaction. But, we still want all parties involved to be certain the swap was conducted fairly and without being influenced by anyone's want for economic gains.
Pseudo random data can be created by our swap engine by simply using inbuilt random functions in the programming language and make use of this as part of the swap transaction. The reason for calling this a pseudo random approach is that not all parts of this generation are provably random. The random function of the programming language might seem random, but is it provably random? In some languages it might be documented as truly random, but in others not. We therefore view this as pseudo random because it is not known. If we end up making use of pseudo random generation, we will at least make sure it generates data which does not follow a clear and observable pattern that can be predicted. If prediction is possible, the functionality would become exploitable.

Our swap engine’s impartial nature could also be questioned as it is set up and operated by the NFT project owner. As we will discuss below, the approach we select will determine if it is provable that the swap engine is behaving impartially and not in the interest of the pool owner.
To have a truly random approach we would need the random data to be generated by an impartial actor using a function that creates provably random data.

Approach 1: Random NFT selected by the dApp and reusing the existing swap pool contracts

This alternative would be a minimum approach where we keep the swap smart contracts developed in phase 1 and add a randomization function to the dApp so that the user (hereafter called ‘Alice’) is allowed to select which NFT she wants to swap, but she is not able to select which NFT to get in return as this is done for her by the dApp. The dApp creates a swap transaction and selects a random NFT to return back to Alice for her selected NFT.

1. Alice explores the swap pool NFTs and sees a lot of NFTs she doesn’t have but would be happy to receive for one of her NFTs belonging to the same policy id
2. Alice connects her wallet to the swap pool dApp
3. Alice clicks a button to initiate a random swap
4. Dialog with all of Alice’s NFTs with the relevant policy id is displayed
5. Alice selects the NFT she wants to send to the swap pool
6. dApp creates a transaction that swaps Alice’s NFT for a randomly selected NFT from the pool
7. Alice signs and submits the transaction
8. The NFTs are swapped on the Cardano blockchain
9. Alice's old NFT is now in the swap pool and Alice has received a new random NFT in her wallet

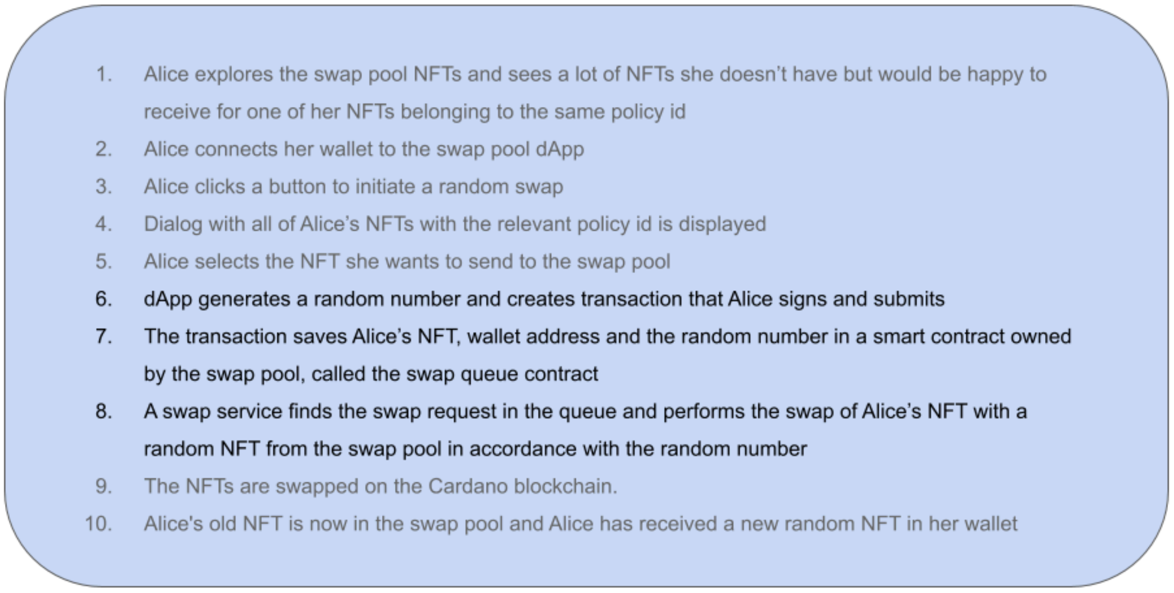
Pros
1. Swap is done in one tx
2. Does not require any additional smart contracts than we currently have to work
3. Random selection is performed by the dApp. Swap pool simply validates legal NFTs are swapped.

Cons
1. Alice can reject the swap if the randomly selected NFT is not to her liking
2. Developers could game the swap solution by creating the swap transactions to select an NFT of their own choice, making it a specific swap.

Because Alice is able to reject swaps if the received NFT is not to her liking, the randomness is highly questionable in this approach. This happens because she is able to inspect the transaction she is about to sign and since the returned random NFT is listed, she can choose to reject it. She can then swap again at a later time and continue to reject until she is content with the received NFT. Even though the NFT returned is random, the end result is not random as Alice is only signing the transaction when she is content with the result.

Approach 2: Random number generated by the dApp, but the swap is performed and NFT returned in a later transaction

In this approach, we have split the swap into two transactions in an effort to remove the possibility for Alice to reject swaps where she is dissatisfied with the NFT she receives in return. To make the difference with the approach above more visible, the common steps are gray, while the new steps are in black



1. Alice explores the swap pool NFTs and sees a lot of NFTs she doesn't have but would be happy to receive for one of her NFTs belonging to the same policy id
2. Alice connects her wallet to the swap pool dApp
3. Alice clicks a button to initiate a random swap
4. Dialog with all of Alice's NFTs with the relevant policy id is displayed
5. Alice selects the NFT she wants to send to the swap pool
6. dApp generates a random number and creates transaction that Alice signs and submits
7. The transaction saves Alice's NFT, wallet address and the random number in a smart contract owned by the swap pool, called the swap queue contract
8. A swap service finds the swap request in the queue and performs the swap of Alice's NFT with a random NFT from the swap pool in accordance with the random number
9. The NFTs are swapped on the Cardano blockchain.
10. Alice's old NFT is now in the swap pool and Alice has received a new random NFT in her wallet

Pros

1. Alice will in practice not be able to foresee which NFT she will receive.
2. Swap is non-deterministic for non-developers.
3. Does not require an oracle to work.
4. Can be solved using the already existing swap contract if we do not require the contract to validate that the swap is done according to the random number. If validation checks a random number to validate, we need new contract logic.

Cons

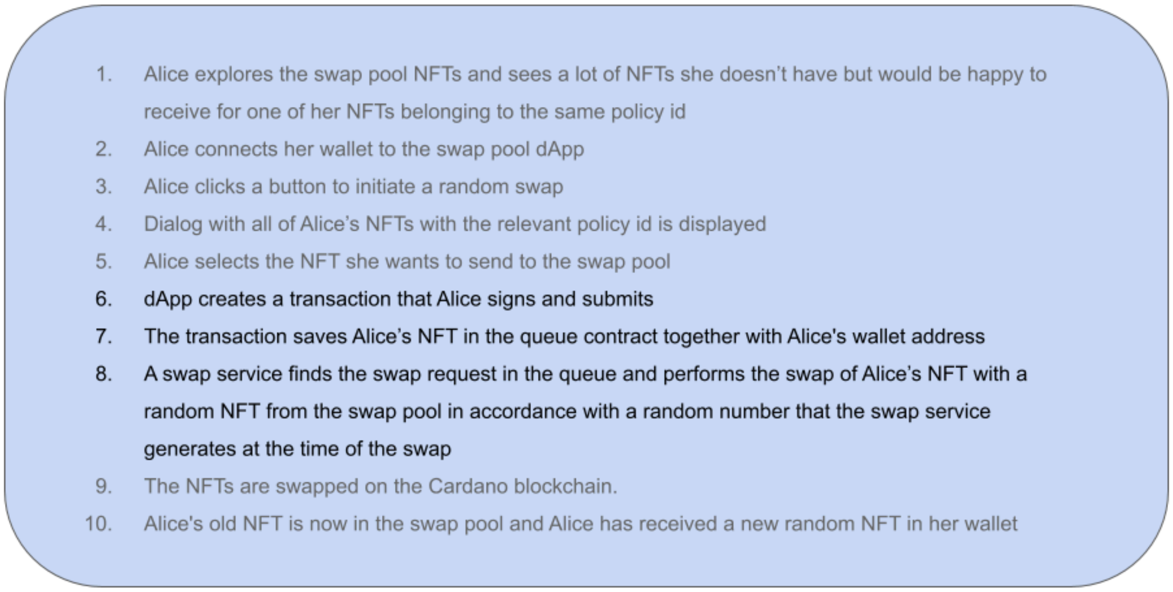
1. Swap requires two transactions to be done => costs more
2. As the random number is generated and saved with the NFT, if Alice is a developer and the random swap is only pseudo non-deterministic, Alice could theoretically, however

unlikely, reject the swap if being able to summize which NFT will be received based on examining the random number and the NFT is not to her liking
3. If reusing the specific swap contracts, developers could game the swap solution by creating the swap transactions to select an NFT of their own choice, making it a specific swap.

This approach definitely has more randomness built into it now that Alice is unable to see which NFT she receives when signing the transaction. If Alice is a developer and is able to see a pattern between the generated number and which NFT is returned, in similar cases historically, she could explore the information on chain and decide to Cancel the swap and thereby getting her old NFT back if canceled in time. This solution is on the other hand non-deterministic when done by non-developers. But as can be seen, this solution still has some challenges when faced with developers

Approach 3: Random selection is done by the swap server function as part of the transaction that returns the random NFT to Alice

The following approach moves the random selection to the last transaction making it difficult to game even for developers as none of the randomness information is available before the transaction happens and can therefore not be predicted. If also changing the swap contracts to allow that only the swapping service can swap the NFTs, then this approach is good enough for most use cases



1. Alice explores the swap pool NFTs and sees a lot of NFTs she doesn't have but would be happy to receive for one of her NFTs belonging to the same policy id
2. Alice connects her wallet to the swap pool dApp
3. Alice clicks a button to initiate a random swap
4. Dialog with all of Alice's NFTs with the relevant policy id is displayed
5. Alice selects the NFT she wants to send to the swap pool
6. dApp creates a transaction that Alice signs and submits
7. The transaction saves Alice's NFT in the queue contract together with Alice's wallet address
8. A swap service finds the swap request in the queue and performs the swap of Alice's NFT with a random NFT from the swap pool in accordance with a random number that the swap service generates at the time of the swap
9. The NFTs are swapped on the Cardano blockchain.
10. Alice's old NFT is now in the swap pool and Alice has received a new random NFT in her wallet

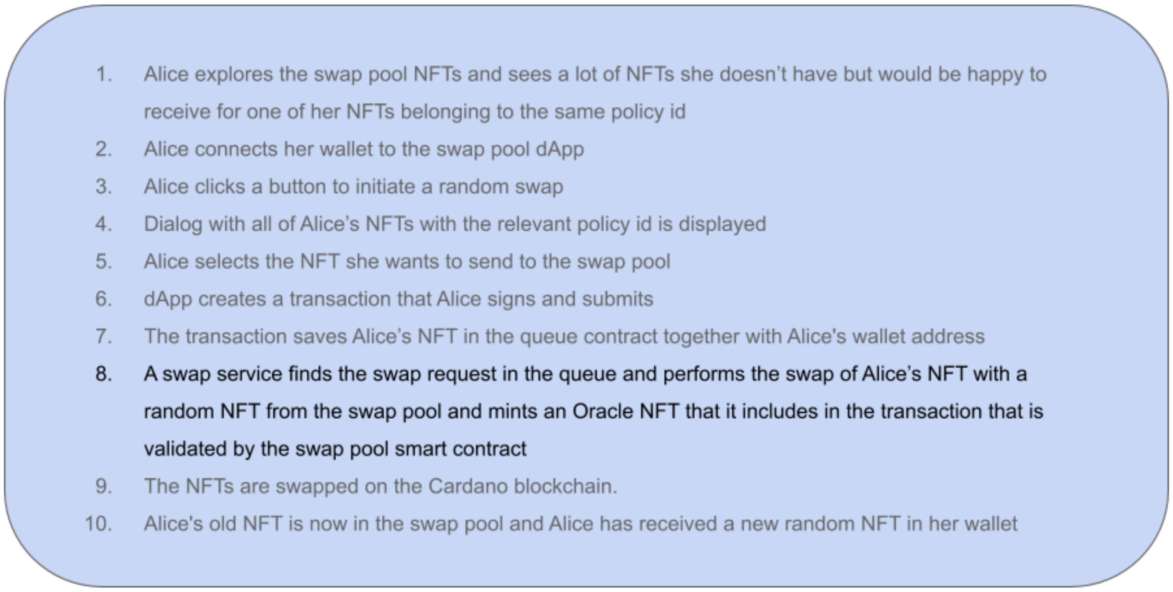
Pros

1. Alice will not be able to foresee which NFT she will receive as the randomness is not possible to examine before the swap is done
2. Does not require an oracle to work.

Cons

1. Swap requires two transactions to be done => costs more

Approach 4: Random selection is done by the swap server function as part of the transaction that returns the random NFT to Alice. An oracle NFT is minted and used by the smart contract when validating the swap



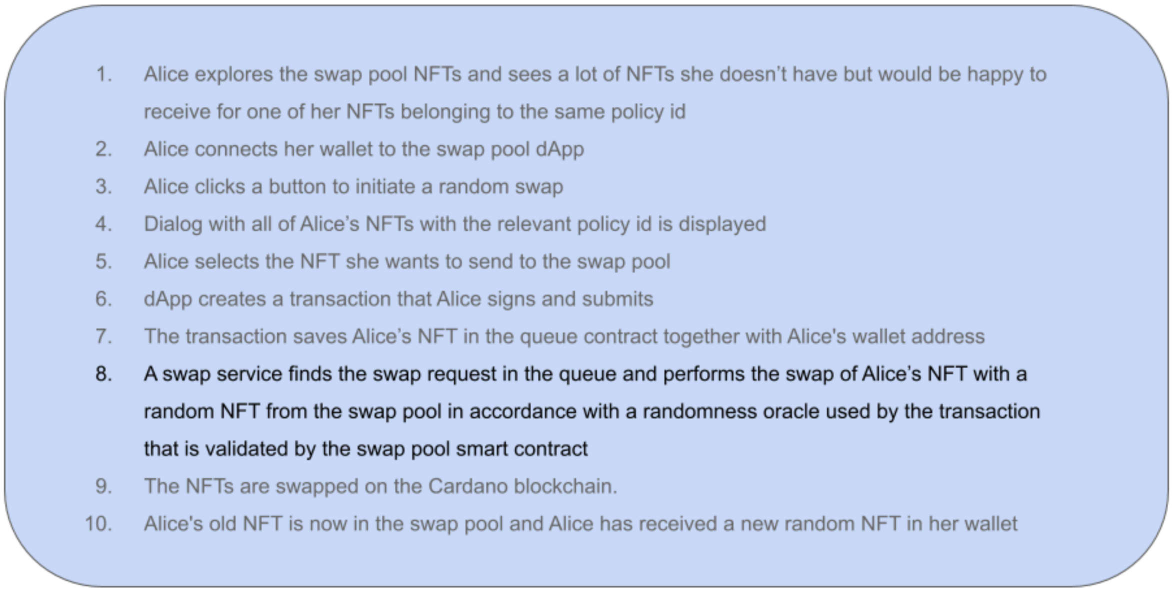
1. Alice explores the swap pool NFTs and sees a lot of NFTs she doesn't have but would be happy to receive for one of her NFTs belonging to the same policy id
2. Alice connects her wallet to the swap pool dApp
3. Alice clicks a button to initiate a random swap
4. Dialog with all of Alice's NFTs with the relevant policy id is displayed
5. Alice selects the NFT she wants to send to the swap pool
6. dApp creates a transaction that Alice signs and submits
7. The transaction saves Alice's NFT in the queue contract together with Alice's wallet address
8. **A swap service finds the swap request in the queue and performs the swap of Alice's NFT with a random NFT from the swap pool and mints an Oracle NFT that it includes in the transaction that is validated by the swap pool smart contract**
9. The NFTs are swapped on the Cardano blockchain.
10. Alice's old NFT is now in the swap pool and Alice has received a new random NFT in her wallet

Pros

1. Alice will not be able to foresee which NFT she will receive as the randomness is not possible to examine before the swap is done
2. Oracle NFT is proof that the swap is done by an authorized party
3. Oracle NFT opens possibilities for new use cases like trophies and proofs for having used the swap functionality. If the Oracle NFT is sent to Alice as part of the swap it can be used as:
    - Proof that Alice has made use of the swap platform
    - Airdrops to supporters of the project or other campaigns
    - Provide services only to users that own the Oracle NFT

Cons
1. Oracle (with accompanying NFT) needs to be minted
2. Swap requires two transactions to be done and also to mint an NFT to place in the Oracle => costs even more
3. New contracts must be developed that examine the oracle

Approach 5: Random selection is done by the swap server function as part of the transaction that returns the random NFT to Alice. The randomness is handled by using an external randomness protocol



1. Alice explores the swap pool NFTs and sees a lot of NFTs she doesn't have but would be happy to receive for one of her NFTs belonging to the same policy id
2. Alice connects her wallet to the swap pool dApp
3. Alice clicks a button to initiate a random swap
4. Dialog with all of Alice's NFTs with the relevant policy id is displayed
5. Alice selects the NFT she wants to send to the swap pool
6. dApp creates a transaction that Alice signs and submits
7. The transaction saves Alice's NFT in the queue contract together with Alice's wallet address
8. **A swap service finds the swap request in the queue and performs the swap of Alice's NFT with a random NFT from the swap pool in accordance with a randomness oracle used by the transaction that is validated by the swap pool smart contract**
9. The NFTs are swapped on the Cardano blockchain.
10. Alice's old NFT is now in the swap pool and Alice has received a new random NFT in her wallet

Pros
1. Alice will not be able to foresee which NFT she will receive as the randomness is not possible to examine before the swap is done
2. Reuse of an existing randomness protocol instead of creating a proprietary Oracle
3. Randomness is generated by a service not operated by the swap pool owners

Cons
1. Swap requires two transactions to be done => costs more

Some words about the randomness protocol

In approach 5, we have identified an external randomness protocol, STEAK, which is decentralized and operated by multiple impartial entities that are incentivized for generating and publishing random data to the Cardano blockchain. The specification of this protocol can be read in the white paper.

In short: The random data is generated using a VRF (Verifiable Random Function) in the same way as normal Cardano stake pools operate already. Entities are registered as so-called STEAK pools and are selected by the protocol (based on pool stake amount) to mine blocks containing a random signature. These blocks are saved on the Cardano blockchain. Because it is unpredictable which STEAK pool (entity) gets a certain block allocation, it is unpredictable what data is used when generating the signature. *This protocol therefore fits our needs perfectly as the data is verifiably random and saved on chain by a random entity which is incentivized for playing by the rules.*

Generating random numbers

If we are unable to make use of the STEAK protocol as we envision for approach 5, we will need to generate the random data and operate the number generation in a way that harmonizes with the trustworthy and fairness properties we need the NFT swap pool to operate according to. Approaches 1, 2 and 3 all have some difficulties regarding the needed trustworthy properties we want the solution to have. For approach 3 we can mitigate most of these reservations by referring to the open source code of the swap functionality to show that the smart contracts only allows the swap when receiving a random number from our swap service and also show that the swap service runs impartially and fairly generates random data.

When generating random data for use in approach 3 and 4, we will make use of one of the following mechanisms to generate it for us:

1. Random.org has been operating since 1998 and provides different services to get truly random numbers based on atmospheric noise. There is also an API that can be used for this purpose: https://www.random.org/clients/http/
2. VRF - We will look into generating the random number / data using inbuilt functionality provided with the Cardano node or other Cardano libraries like Cardano serialization library
3. Randomness libraries designed to generate random data that can be incorporated into projects;
    a. https://www.npmjs.com/package/seedrandom
    b. https://github.com/ckknight/random-js
    c. …more libraries available on Google
4. Programming language inbuilt Math.random() function.

We will look into the top 3 mechanisms mentioned in this list and make use of the functionality that first delivers the desired random data in an unpredictable manner. All these sources will deliver randomness according to the desired properties of our project as they are all designed for (close to) truly random data generation. Mechanism 4 is our last resort and only listed here for completeness.

Project design decisions

This project is aiming at implementing the random swap in accordance with either approach 4 or 5. Approach 5 is the vision, but has some inherent risks attached. The STEAK protocol is new and has had some difficulties in the startup and is currently being fixed. The highest risk is that this protocol will be unsuccessful and not be in operation because of not attracting enough entities to run a STEAK pool. The design is decentralized, and so there is not one owner that can take this decision which is a good thing. As long as the protocol is able to attract pool operators, this protocol will continue to add random data oracles to the Cardano blockchain. Approach 4 is the fallback approach if we do not see the STEAK protocol succeed within a couple of months. We do not want to create a proprietary random oracle solution as we want to support and make use of already running protocols when possible. Success for STEAK is defined as being operated by at least 3 entities and that it publishes random data to the Cardano blockchain at the interval defined by the protocol.