A Protocol for NFT Migration

GUILLAUME GONNAUD AND EDOUARD BESSIRE

February 2021

/!\ This Protocol was drafted as part of a Web3 Foundation grant, and is currently being implemented in the myNFT bridge. We encourage the community to add their comments to this document, test the myNFT bridge and reach out to us at bridge@mynft.com

1. Abstract

Assets of all kinds will soon be tokenized and traded as non-fungible tokens, fostering the need for a standard to migrate and track the ownership of these assets across universes. Current NFT migration projects focus on combining the bridge and a trust-minimized relay together in an exclusive way, which has limitations. We propose a model where the bridge only acts as an escrow and messaging service, whilst relays are chosen in consensus by both token creators and token owners, and are responsible for the state transmission. Thanks to this architecture, there needs to only be one bridge per universe, and only as many endpoints as the number of universes to migrate to. The bridge itself is fully decentralized and trustless, and anyone is free to create a relay to perform migrations using that bridge. This also makes the work of tracking NFT provenance across universes much easier for applications. The protocol standardizes the procedure and the information sufficient and/or necessary for the migration of an ERC-721 compatible asset from one universe to another, as well as the data structure for cross-universe NFT tracking. Some implementation issues are considered...

2. Introduction

2.1 Motivations

Starting with the publication of the Bitcoin white paper [1] in 2008, a new form of digital ownership emerged. Designed as an electronic alternative to cash, the provable digital scarcity provided by the blockchain served to produce a limited amount of fungible and divisible Bitcoin

tokens. Consequently, this model was employed on other blockchains with smart contracts to use tokens not only as means of exchange, but also as units of account, stores of value, and as pseudo-securities [2]. The creation of non-fungible tokens in 2017 with the launch of CryptoPunks [3] on the Ethereum blockchain marked yet a new paradigm shift for digital ownership. Non-fungible tokens are indivisible, identifiable and thus their history and provenance are traceable. With these properties, tokens can represent ownership over digital or physical assets. Whereas Bitcoin was designed as a peer-to-peer electronic cash system, non-fungible tokens laid the foundation for a peer-to-peer electronic asset ownership system.

Tokenizing an asset by representing its ownership as a non-fungible token on a public blockchain makes it publicly tradable with instant settlement worldwide, and provides a transparent and public record of ownership. Depending on the asset class being tokenized, it could also make it much cheaper to trade and increase its liquidity. In the last few years, we have seen tokenization begin to disrupt the art and collectible market, as content creators started tokenizing their digital creations, turning them into scarce, tradable assets and disrupting the traditional advertising and subscription-based business models for digital content [4]. Tokenization has the potential to inspire new business models and unlock significant value in many existing asset classes and untapped markets. The world is about to be tokenized, and unique assets of all kinds will soon be traded as non-fungible tokens.

Tokenizing an asset means bridging the universe (e.g. the real world, a video game company's private database, a blockchain, a parachain) of the asset, with the universe where the token representing that asset is going to be. The liquidity of tokenized assets ("NFTs") is also today limited by the fragmentation of blockchains, and as NFTs become more prevalent, this fragmentation will only increase, fueling the need for bridges between blockchains as well. Without a consensus on how ownership of an asset is represented across universes, this ownership cannot be verified. A standardized, consensual cross-chain NFT ownership protocol, allowing to bridge those tokens from one universe to another, is needed. As a standard for decentralized NFT ownership, this protocol and the associated bridges must be open-source; any kind of intellectual property rights attached to a bridge would endanger the legal grounds of NFT ownership. Current bridges are focusing on trust-minimized state transmission, which is satisfying for fungible, divisible tokens, but not for assets tokenized with non-fungible tokens.

2.2 Objective

The aim of this standardization is to identify which actor needs to trust what components in an NFT cross-chain migration, to define a way to carry out NFT migrations, and to be as generic as possible in the components specific implementation.

More specifically, the protocol must allow for :

- Tracking the provenance of an NFT whose history spans several tokens across multiple universes.
- Achieving the transmission of ownership of an NFT from one universe to another.

- Clearly separating a deed to the NFT from the NFT itself and from their technical representation as tokens.

Moreover, this standard needs to be generic enough so that migrations are not limited to blockchains, but can also be performed on compatible APIs.

2.3 Scope

This specification is intended for any universe and API that is compatible with the ERC-721 standard [5], an open standard that defines a minimum interface a smart contract must implement to allow non-fungible tokens to be managed, owned, and traded on the Ethereum blockchain. ERC-721 has become a de-facto standard for NTFs on the Ethereum blockchain and it is sufficiently generalist[1] to be abstracted and used as a standard across other universes and to define the scope of the NFT migration protocol. Hence, our protocol, like the ERC-721 standard, is not restricted to EVM-compatible blockchains, but extends to all universes that support cryptographic signatures, or at least a reasonable proof of identity that can be validated by the destination universe.

This standard is primarily made for:
- ERC-721 compatible NFT publishers and applications
- Relay operators
- Marketplace developers
- Wallet developers

NFT owners should not have to read the standard themselves, as NFT provenance (successive list of owner/migrations) should be presentable as clear graphical user interface information to the user in the relevant apps.

[1] The ERC-721 standard does not mandate a standard for token metadata or restrict adding supplemental functions.

3. Terms and Abbreviations

This section lists and defines the terms and abbreviations used throughout this specification.

3.1 NFT

An asset that is compatible with the ERC-721 standard. In this definition, the asset is abstracted from the non-fungible token used to represent this asset on a blockchain. An NFT always exists in a world, which is itself in a universe. Examples of NFTs include a piece of digital art, a domain name, a video game skin, a concert ticket...

3.2 Universe

A universe is a set which follows the ERC-721 standard, and contains a set of worlds. Examples of universes include blockchains, parachains, the Web2.0 internet, a private company's video game.

3.3 World

An element of a universe which contains a set of tokens, and has an owner (see World Owner). Examples of worlds include a smart contract or smart contract ecosystem, a website, a private company's project...

3.4 Token

An element of a world which can represent the ownership of an NFT, and has an owner (see Token Owner). A token is identified by the universe and the world it is in, as well as a token identifier (a byte array) which is unique to the world the token is in.

t = t (U, W, i) where U is a universe, W a world with $t \in W$ and $W \in U$, and i the token's unique identifier in W.

If two tokens are identified as $t_1 = t(U_1, W_1, i_1)$ and $t_2 = t(U_2, W_2, i_2)$
Then, $t_1 = t_2 \Leftrightarrow \{ U_1 = U_2 ; W_1 = W_2 ; i_1 = i_2 \}$

3.5 Representative Token

A token that represents the ownership of an NFT. Only one token can represent the ownership of a given NFT at a given time. This ownership can be transferred from one token to another via a migration.

3.6 Token Owner

The entity identified as the owner of a token by the world the token is an element of.

The owner of a token should be able to transfer the token to another entity. If they are not, the token is considered “burned”, as it cannot be put into circulation.

If the owner is a wallet, then the owner is able to identify themselves cryptographically by signing a message with their private key.

A token owner can have operators for all tokens they own now and in the future in a specific world (ERC-721 *setApprovalForAll* function).

A token owner can have operators for a single token (ERC-721 *approve* function)

3.7 Operator

An entity trusted by the token owner which can act on behalf of that token owner.

If the operator is a wallet, then the operator is able to identify itself cryptographically by signing a message with its private key.

An operator of a specific token can nominate and remove operators of that specific token.

3.8 NFT owner

The entity that is the owner of the NFT’s representative token.

3.9 NFT Bridge

An element of a universe which implements the bridge interface that is going to be defined in this standard.

The bridge interface allows entities to hold a token in escrow, emit the appropriate events, data and callback for migrations as well as redeem tokens in escrow when the appropriate functions are called during a migration.

3.10 World Owner

A world owner is the entity defined by a world as its owner.

A world owner is assumed to be the rights holder and publisher of tokens in this world.

A world owner can have relays who can act on its behalf.

3.11 Relay

An entity trusted by the owner of a world and a token owner to act on their behalf when reading data from a remote bridge and writing migration data to the local universe bridge.

The relay connects the origin and destination bridge in a migration, and can say if a token can be redeemed after remigration.

A relay can be one of three kinds:

- Trustless: The relay can read the origin bridge and world and write into the destination bridge in a trustless fashion.
- Trust-minimized: incentivization mechanics and financial penalties are put on catching a relay lying.
- Trusted: there is no on-chain mechanics ensuring the relay is truthful. Instead, it relies on the legacy world trust, backed by potential lawsuits that are deemed as deterrent to the relay's lying.

3.12 Event

Message emitted by a world or a bridge that can be read and reacted to by third parties.

The history of an NFT's ownership can be reconstituted by reading events from the bridge and their past and present representative tokens' worlds.

3.13 NFT Migration

The transfer of representation of the ownership of an NFT from one token to another (see Full Migration), or the creation of a deed to the representative token of an NFT (see IOU migration).

Let $t_1 = t(U_1, W_1, i_1)$ be the representative of an NFT and $t_2 = t(U_2, W_2, i_2)$ a token with:

t_1 the origin token
U_1 the origin universe
W_1 the origin world
i_1 the origin token's unique identifier in W_1

t_2 the destination token
U_2 the destination universe
W_2 the destination world
i_2 the destination token's unique identifier in W_2

Note: The origin and destination universe/world of a migration do not have to be different i.e. intra-universe migrations are possible. However, t_1 and t_2 must be different.

O_1 the origin owner of t_1 (before migration)
O_2 the destination owner ot t_2 (after migration)

B_1 the origin bridge
B_2 the destination bridge

We can say that the NFT has migrated from t_1 to t_2 if and only if:

(1) B_1 is the token owner of t_1
(2) O_2 is the token owner of t_2
(3) t_1 is in escrow with B_1 i.e. t_1 cannot change owner unless t_2 is put in escrow with B_2

3.15 IOU Token

A token which represents a deed to the representative token of an NFT.
IOU is a financial term.

3.16 IOU Migration

An NFT migration where the representative token of the NFT does not change. This representative token stays in escrow with the bridge, and a deed to this token is migrated and traded within the migrated world. As a deed, if the IOU token is migrated back to the original token, then the IOU token is locked into B_2 with no way of recovering it.

Once the migration has been completed, NFT ownership has not been transferred. Instead, a deed to the representative token of the NFT has been created.

3.17 Full Migration

An NFT migration which is handled by the NFT's world owner, who controls the destination world, and thus can allow the NFT's features and intellectual property rights to be transferred to the new token in the destination universe.

When technically possible, the same token unique identifier should be used across worlds.

4. Background

This section analyses the current work being done towards NFT migrations, their limitations, and proposes solutions to these limitations.

4.1 State of the Art and Limitations

Efforts have already been made to migrate ERC-20 tokens [6] and other arbitrary data [7] across blockchains. However, there is no consensus yet on how to migrate NFTs specifically. Current NFT bridge projects focus on the development of trust-minimized relays [8]. In this approach, the trust in the relay is built with a bounty system: The relay stakes value (e.g. money) in order to operate on the bridge, and if the relay lies about a migration, the relay loses the value it staked. This financially incentivises the relay to be truthful, but only as long as what is being migrated is worth less than the value staked. In other words, this system puts a price on lying, which is equal to the value staked. This is not a significant issue when migrating divisible tokens (e.g. ERC-20 tokens), as one can always transfer a value lower than this price. But this approach is not suitable for NFTs, because they are not divisible and no one can predict the value of NFTs that will be migrated in the future. Therefore, in the case of NFTs, the trust in the relay cannot be purely monetary-based, and must come from elsewhere. Another limitation of existing NFT bridge projects is that they combine the bridge and relay together in an exclusive way [9] [10]. A likely consequence is that projects will be incentivised to build their own bridge rather than trust someone else's relay. As a result, applications would need to track all bridges endpoints from each universe, with each bridge's endpoint probably having its own API. Finally, work on current bridges has been mostly focused on the technical transfer of data rather than on the transfer of ownership above the technical data. Contrary to fungible tokens, the value of a ERC-721 token comes from its provenance and the intellectual property attached to it. If a technical transmission of state is achieved but the transfer of ownership of the intellectual property is not recognized by applications, then the migration would have de-facto failed. Without the blessing of the token creator and the legal contract linking an NFT asset and features to its representative ERC-721 token, what will be migrated at most is a deed to the asset, not the asset itself.

4.2 Proposed Solution

It is our conviction that, save for bidirectional trustless (i.e. cryptographically secure and computationally impractical to forge) reading and writing of data, there is no "perfect" trust-minimized state transmission possible, only compromises. Therefore, the transfer of NFT ownership should be at the center of the migration protocol, not the minutiae of the technical proof of state transmission incentivization mechanisms. We believe that rather than having the bridge force the choice of relay to the owners and creators, a bridge should instead only act as

an escrow and messaging service while relays are chosen in consensus by both token creators and token owners, who should be the ones responsible for the state transmission. This way, the bridge itself can be fully decentralized and trustless, whilst the elements that require trust from users are left for the user to choose depending on the kind of assets they wish to migrate. Thanks to this architecture, there needs to only be one bridge per universe, and only as many endpoints as the number of universes. This makes the work of tracking NFT provenance across universes much easier for applications.

4.3 Existing NFT bridge projects

Chainbridge
Chainbridge is a technical demonstration showing that transmitting the state of an NFT in a trust-minimized fashion is possible. However, its features are focusing on making trust minimized relays: at no point the NFT publisher is involved, and without the consent of the publisher, those migrated NFTs would be considered counterfeits. Chainbridge is solving what we identify as the relay problem of state transfer. With minor adaptation, Chainbridge's trust minimized flow could be integrated as a relay NFT creators can use.

t3rn
t3rn is a protocol for interoperable code execution between multiple blockchains on Polkadot. Specifically, t3rn focuses on cross-chain swaps and atomic smart contracts. t3rn's mechanism allows for potential trustless cross-parachain communication, and such features will be needed for NFTs. However, NFT provenance would be lost using most of the features for trade offered by t3rn.

Darwinia
Darwinia's cross-chain NFT solution and standard is, in scope, the closest project to our work. The main difference being that Darwinia proposes to solve the trust-minimized relay problem, whilst our line of thinking is that the choice of the relay should be left to the relevant actors (token creators and token owners) and is in fact NOT in the scope of a bridge standardisation.

Snowfork
Snowfork is a bridge/relay that works between Ethereum and Substrate-based parachains on Polkadot. Snowfork is solving what we identify as the relay problem of state transfer. It is a technical solution focused on ERC-20 tokens and not a standard for generic NFT migrations because at no point the NFT publisher is involved, and without the consent of the publisher, those migrated NFTs would be considered counterfeits. With minor adaptation, Snowfork's trust minimized flow could be integrated as a relay NFT creators can use.

RMRK
RMRK is the first unofficial shared library of the Polkadot ecosystem, allowing for seamless, cheap, and direct migration of non-fungible assets across all connected parachains with just

XCM. RMRK's solution does not solve the EVM -> Substrate issue, neither does it have any solution for non-parachains, but should be mentioned as a bridging method that is in progress for Substrate based chains specifically. Ideally this protocol and RMRK protocol will evolve toward compatibility with each other.

5. Specifications

This section specifies measurable features that the NFT migration protocol must support, as well as how to test these features using pseudocode.

Minimum viable NFT that can be migrated
-Owner who can prove it exist
-Sign messages
-Capacity to send to bridge
-Capacity for the owner to set the Bridge as the owner of the token

Additional features that need to be added :
Allows for SODA relays and Dapps to read and interact with the bridge

5.1 Migrate any ERC-721 token between ERC-721 compatible universes

Feature 1	The bridge should allow any ERC721 token to be put in escrow in the origin universe.
Test 1	Any arbitrary ERC721 token supporting safeTransfer can be sent to the bridge for migration.

•

Feature 2	As long as the token is in escrow, it cannot be transferred to anyone else in the origin universe.
Test 2	As long as the token is in escrow (migrated), all call to transfer/safeTransfer on the token will fail.

•

Feature 3	The destination bridge could be made to receive an NFT from any ERC-721 origin universe with a bridge. This allows for complex IOU/Full migration token behaviour. The bridge is just a recordkeeper of the migration.
Test 3	The bridge can associate an arbitrary ERC-721 token it is the owner of (but that was not put in escrow) with incoming migrations data. This migration data is then stored in the bridge and readable.

•

Feature 4 A successfully migrated token is sent from the bridge to its destination owner.

Test 4 At the end of the migration process, a migrated token owner is the owner that was specified in the original migration call on the original chain.

5.2 The token owner can get the original token back when migrating an NFT back and forth (Reversibility)

Feature 5 A token put in escrow should be transferable again if migration data coming from the destination token comes back to the original bridge and designates the original token as the destination token's destination.

Test 5 If an NFT is migrated to a destination universe and then back to the original universe with the same original token as a migration target, then the original token is removed from escrow and given to the new owner.

Comment 5 If an NFT is migrated from $t_1 = t(U_1, W_1, i_1)$ to $t_2 = t(U_2, W_2, i_2)$, then the owner of t_2 must be able to migrate the NFT back from t_2 to t_1.

5.3 Choose between and perform an IOU Migration or a Full Migration

Feature 6 Perform an IOU Migration (no Digital Rights Management requirement)

Test 6 An NFT can be migrated from t_1 to t_2, with t_2 being an IOU of t_1.

•

Feature 7 Perform a Full migration, which requires Digital Rights Management at the bridge level. When such Full Migrations are happening, callback hooks can be set up by NFT creators and publishers so that they can execute in-universe relevant code.

Test 7 An NFT can be migrated from t_1 to t_2, with t_2 becoming the representative token of the NFT.

•

Feature 8	Choose between an IOU migration or a Full Migration
Test 8	Only the owner of a token has the final word on whether to perform an IOU or a Full migration. Other actors, including token creators, can only provide the option to do so.

NFT creators and publishers need to be confident that the bridge is not going to be used to create counterfeits of their NFT, or anything that removes or damages the license they attach to the NFT. Hence, any migration that is not approved by the NFT creator/publisher is only possible as an IOU. NFT creators and publishers can specify destination worlds for full migrations that implement all of the features that the original token has in the original world (e.g. siring Cryptokitties [11]).

5.4 Be ERC-1155 [12] compatible

Feature 9	All functions, events and message should be able to either accept ERC-721 or ERC-1155 tokens
Test 9	Escrow and migrations should be possible using *safeBatchTransferFrom()* to transfer multiple ERC-1155 tokens at once, with the same restrictions and features as safeTransfer

5.5 A migration must allow for trustlessly verifiable transfer of additional data specified by the original owner as the last argument of *safeTransfer(..., bytes[])*, which is called when transferring the migrated token to the destination owner.

/!\ We need feedback on the below from the community

Feature ? The standard must provide a way for the owner to specify a bytes array to be called when the destination token is being transferred to the destination owner. What is proposed is to directly allow for the extra bytes array to be set when depositing the token into the bridge.

Feature 10 When a token is put in escrow using the safeTransfer(..., bytes[]) method with arbitrary bytes[] as the last argument, then the destination bridge should call safeTransfer(..., bytes[]) with the same arbitrary bytes[] as the last argument when releasing it to its owner in the new universe.

Test 10 If the token was put in escrow with safeTransfer(..., 0bxxxx), then the only way the token is released to it's new owner is with a safeTransfer(..., 0bxxxx) call containing the same binary sequence. Other binary sequence or empty binary content will fail to release the token.

5.6 The original token owner can specify any destination world (destination worlds are not exclusive per universe)

Feature 11 Only the owner of a token can decide where a token is migrated with an IOU, and it can be to any arbitrary IOU world. Minting IOU tokens is not exclusive to any company/smart contract in the destination universe. It also means it's the owner's responsibility to ensure the migration can succeed or that the IOU tokens minted accurately represent ownership. If not, the original token would be effectively stuck in the origin universe bridge (burned). Full migration destination universes and worlds need however to be whitelisted beforehand, in order to prevent counterfeits.

Test 11 If a token from origin universe U and world W can be IOU migrated to an arbitrary destination universe γ world α, then it should also be able to be migrated to a different arbitrary destination IOU world β within universe γ, at the sole token owner's discretion.

5.7 Migration paths need to be pre-registered

Feature 12 In order to prevent accidental bad migrations, any world that is accepted as a destination for IOU tokens needs to be registered beforehand in the origin bridge as well as the destination bridge.

Test 12	Migrations using an unregistered migration path should fail. Anyone creating an IOU minting world should be able to register it as a valid destination target in the relevant destination and origin bridges.

5.8 Allow trustless NFT migration between Polkadot [13] parachains using SPREE.

Feature 13	More generally speaking, if a trustless oracle exists between the two universes, it should be used to minimize the amount of trust necessary.
Test 13	Can migrate between two parachains (See RMRK work and make it part of the standard/adapt the standard ?)

A chain that is writing NFT migration data in the relay chain in Kusama should be able to migrate it to the Polkadot network.

5.9 Intra-universe Migration

Feature 14	The bridge must be usable within the same universe and or world. t_1 and t_2 must have different token identifiers.
Test 14	t_1 can be migrated to t_2 even in the case $U_1 = U_2$ and $W_1 = W_2$.

5.10 Time expiration on IOUs

Feature 15	When the expiration timestamp is reached, the destination IOU token becomes unable to redeem the origin token in escrow in the origin universe. Similarly, when the expiration timestamp is reached, the origin token can be transferred out of the origin bridge after a safety period (to prevent redeeming before the IOU expiration).
Test 15	...

This feature is to accommodate for ERC-809 [14], ERC-1201 [15], and other time gated NFT standards.

ERC-809: Renting Standard for Rival, Non-Fungible Tokens
https://github.com/ethereum/EIPs/issues/809

ERC-1201: Two Tiered Token Structure for Non-fungible Asset Ownership and Rental Rights
https://github.com/ethereum/EIPs/issues/1201
Introduce validity duration for tokens.
Not a problem for Full migrations, but have implications for IOUS and marketplaces...

The term timestamp is used, but any height property compatible with both the origin and destination chain (number of blocks, etc) can be used. The relays are responsible for ensuring consistency.

5.11 safeTransfer equivalency for non-EVM chains

Feature 16	Allow for native "safeTransfer" equivalent in the Migrates functions. If the blockchain is implementing natively NFTs that are not ERC721, then the bridge MUST allow for these tokens to be migrated to and from the bridge.
Test 16	A bridge should be possible for WASM/substrate based parachain (like current NFT projects on Kusama)

5.12 Integration of other standards (Discussion)

ERC-1633: Refungible
https://github.com/ethereum/EIPs/issues/1634
Is very ecosystem dependent. Individual ERC20 can be migrated, IOU can be migrated, but a proper refungible token will need a full migration

ERC-994: Delegated Non-Fungible Token Standard
https://github.com/ethereum/EIPs/issues/994
to delegate ownership and control
No issue for either IOU or full migrations. Those are simply token with additional, on chain metadata. Special care needs to be taken when migrating those tokens.

Such token are going to be VERY framework dependent depending on how much is on-chain or offchain and do not specially need extra protocol features for migration as fullMigration is going tobe necessary for full features.

ERC-998: Composable Non-Fungible Token Standard
https://github.com/ethereum/eips/issues/998
compose NFTs into hierarchical ownership schemes
The very concept of migration is incompatible with ERC998ERC721, as the bound assets need to be migrated separately.

No issue for ERC998ERC721.

6. Assumptions and Migration Model

6.1 The Game Theory Behind Bridges

1 Incentivization and bounty mechanics only put a price on lying. Therefore, the NFT migration protocol should not specify any trust minimization mechanism for the relay themselves.

2 If two separate universes have independent consensus protocols and no trustless communication channel between them, it is not possible to achieve a trustless consensus on data. Therefore, the relay that migrated the NFT needs to be trusted by the token owner.

3 The owner of a token trusts the creator of the token that this token is representative of an NFT. If not, they would not be owning the token.

4 The creator of a token does not want this token to be considered a counterfeit or be distributed to the wrong owner. As a result, the creator of a token can be trusted to be a relay to migrate an NFT toward this token, and the token owner trusts the token creator to be a relay.

5 The creator of a world will want to be able to designate relay operators they personally trust to handle token migration, rather than handling the migrations themselves. Those operators could themselves be following trust minimized mechanics. This bridge protocol is agnostic on how exactly trust minimization is achieved. Only the trust of the token creator and of the token owner in those relay operators matters.

6 As the token creator trusts relays other than themselves and the token owner trusts the creator, therefore the token owner trusts the relays chosen by the token creator. **This solves point 2 above.**

7 Only the owner of a token should decide if the token is migrating or not using the bridge, which relay is used for the migration, and to where it is migrating. Other actors should only provide choices, not force actions upon the token owner.

8 Token creators might want to only allow migrations toward/from a specific world (exclusivity)

6.2 Migration Model

Based on the assumptions above, we propose the following model for the NFT migration protocol:

- Token creators can designate and remove trusted relays in the bridge to handle migrations of their tokens.
- Token creators can designate specific worlds in other universes for their tokens to migrate with full migrations. A callback at the end of the migration process allows the token creators to execute any necessary code on both the origin world and the destination world.
- Original Token owners need to personally sign a proof of deposit in escrow in the origin bridge for the migration to proceed
- Original Token owners need to personally sign a proof of migration (before destination token given to destination owner)

7. Migration Procedure

This section details the procedure for a representative token owner to migrate an NFT, either as an IOU or as a new representative token.

Instructions are written in pseudocode rather than any specific language. Any and all arguments are optional if they can be recovered from the environment (msg.sender, method in the origin or destination world, oracle palettes in a parachain, etc...)

Prerequisite for a NFT Migration to start

Let t_1 representing an NFT and t_2 be two tokens with $t_1 = t(U_1, W_1, i_1)$ and $t_2 = t(U_2, W_2, i_2)$

t_1 the origin token
U_1 the origin universe
W_1 the origin world
i_1 the origin token unique identifier W_1

t_2 the destination token
U_2 the destination universe
W_2 the destination world
i_2 the destination token unique identifier in W_2

O_1 the owner of t_1 i.e. the origin token owner
O_2 the owner ot t_2 i.e. the destination token owner

B_1 the origin bridge
B_2 the destination bridge

In order to start an NFT migration, the following assertions must be true:

(1) The destination token is owned by the destination bridge
(2) A relay that can write on the destination bridge exist (accredited by destination publisher)

Step 1: Pre-Registering the migration

O_1 or an operator callss B_1.*_migrateTo_(O_1, W_1, t_1, U_2, B_2, W_2, t_2, O_2), with the *_migrateTo_() function being either _IOUmigrateTo_() or _fullMigrateTo_(), to announce the intention to perform a migration.

The transaction will fail if the caller is not an operator or the owner of the t_1.

Step 2: Putting the origin token in escrow

O_1 or an operator calls W_1._safeTransferFrom_(O_1, B_1, t_1, _data_), putting t_1 in escrow with B_1.

This transaction will fail if the migration was not pre-registered (see Step 1).

This transaction locks the token and makes it only recoverable in the case of an NFT migration from t_2 to t_1 (the inverse migration). If the migration from t_1 to t_2 was an IOU migration, then the inverse migration would lock t_2 in escrow in B_2

At the end of the safeTransfer, a Callback set up by the token's world owner is called if the migration is a full migration

This transaction generates a **proofEscrowHash** (made public through an event), which depends on the block the transaction is mined in and the transaction parameters.

Step 3: Signing proofEscrowHash

O_1 then needs to sign **proofEscrowHash**, generating **proofEscrowHashSigned**.

Step 4: Writing migration data in the destination bridge

Any accredited relay can call B_2._migrateFrom_(U_1, W_1, t_1, O_1, B_1, W_2, t_2, O_2, _blockMined_, **proofEscrowHashSigned**, _data_).

Due to technical limitations on stack sizes, several calls to services functions might be necessary to fill the data argument properly beforehand.

If the caller is not an accredited relay by the owner of W_2, then the call will fail.

Subscribing to a relay collection, necessary in the case of some trust minimized paradigm is possible simply by having a smart contract being the relay.

In universes where a relay can be a trustless truth provider, then this truth provider is used (so a call to *DB.MigrateFrom* would check the truth provider and a lot of arguments not necessary)

This call will generate a *migrateFrom*(*bytes[]* **migrationRelayedHash**) event.

Every relay that calls this function will be added to the list of signee of the migration, each generating a different **migrationRelayedHash**.

Step 5: Origin owner validating the migration data

O_1 selects the relay that they trust to have properly written the migration data in the destination bridge by signing that relay's **migrationRelayedHash** and makes it public as **migrationRelayedHashSigned**.

Step 6: Transferring the destination token to the destination owner

Anyone can call B_2.*finalizeMigration*(**proofEscrowHash**, **migrationRelayedHashSigned**).

This will make B_2 call W_2.*safeTransferFrom*(B_2, O_2, t_2, *data*) according to the data that generated **proofEscrowHash**.

At the end of the safeTransfer, a Callback set up by the token's world owner is called if the migration is a full migration

Requirements for a NFT migration to be completed

An NFT migration is considered completed when the following assertions are true:

(1) B_1 is the token owner of t_1
(2) O_2 is the token owner of t_2
(3) t_1 is in escrow with B_1 i.e. t_1 cannot change owner unless t_2 is put in escrow with B_2 and the NFT is migrated back to t_1

8. Discussions

Limitations

The destination bridge needs to be able to recognize the origin owner's signature to validate the migration data <= Each pair of destination/origin universes needs a specific verification procedure. EVM to EVM is easy, but other cases need to be standardized. This also implies the bridge will need a governance system for upgradability.

Data[] exploit with hashes. => Each signature verification needs to be aware that Data[] could be manipulated by the relay, and hence the hash needs to be computationally impractical to reverse.

Requirements

-Universe names/references for migration need to be standardized. We propose human readable strings instead of an ENUM, which are then put as a single BYTE array. (256bits ?)
-Each universe has their own way of writing world addresses that can fit as an argument in the origin universe. Requirements would be once again a single BYTE array long enough to identify the destination world for each destination universe. (256bits ?)

Basically, each universe pair needs to be standardized

Worst cases exploration

Crooked relay
In the destination bridge, a relay could lie about someone migrating something in the origin universe, and use an address that actually does not own the token to sign the transactions, imbuing arbitrary tokens with ownership. This would make the token counterfeit.
If a token is seen as counterfeit, it has no value anyway. This is not a new problem for marketplaces to remove scams from their listings.
Such a token can be easily detected as a scam if migration data in the destination bridge does not match migration data in the original bridge.
Ultimately, it is up to token creators to carefully select their relays.

User Experience

Total calls :
Original owner need to
-Transact the setup of his token migration <= gas spent, can be done by operator
-Transact the transfer of the token <= gas spent, can be done by operator

- Sign proofEscrowHash
- Sign migrationRelayedHash

Relay needs to :
-Call migrateFrom

Anyone can :
Call finalizeMigration

Someone needs to :
Put the destination token in the bridge (could be done by default by the IOU smart contract itself, allowing the bridge to mint tokens at will)

So all in all, assuming fully subsidized migration, an owner needs to :

-Do ONE transaction that designate an operator on the origin universe
-Click sign twice with his origin universe wallet

9. Acknowledgments

The authors wish to thank a number of contributor for helpful comments during early
Discussions and drafting of the protocol, especially Grigoriy Zaytsev who constructively commented on who should decide what code to run when an NFT is migrated as well as suggesting the addition of time restrictions on IOU tokens; and xxx who commented on xxxx.

Bruno Škvorc, RMRK

The Web3 Foundation, which funds research and development teams building the foundations of the decentralized web, and supported the creation of this protocol via their Open Grant Program.

David Hawig, Head of grants at Web3 Foundation

Alexandre Guillioud

10. References

[1] S. Nakamoto, "*Bitcoin: A Peer-to-Peer Electronic Cash System*", https://bitcoin.org/bitcoin.pdf, 2008

[2] V. Buterin, "*Ethereum Whitepaper*", https://ethereum.org/en/whitepaper, 2013

[3] Larva Labs, CryptoPunks, https://www.larvalabs.com/cryptopunks, 2017

[4] Cryptograph, "*About*", https://www.cryptograph.co/About, 2020

[5] W. Entriken, D. Shirley, J. Evans, N. Sachs, "*ERC-721 Standard*", http://erc721.org https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md, 2017

[6] PolkaBridge, "*Bridge between Polkadot platform and other blockchains.*", https://polkabridge.org, 2020

[7] Snowfork, "*Polkadot Ethereum Bridge*", https://github.com/Snowfork/polkadot-ethereum, 2020

[8] Darwinia Documentation, "*Darwinia Cross Chain Nft Bridge Protocol*", https://docs.darwinia.network/docs/en/rfc-0010-darwinia-cross-chain-nft-bridge-protocol, 2020

[9] Moonbeam Docs, "*ChainBridge's Ethereum Moonbeam Bridge*", https://docs.moonbeam.network/integrations/bridges/ethereum/chainbridge, 2020

[10] B. Maciej, "*t3rn: Protocol for blockchain interoperability*", https://github.com/t3rn/t3rn, 2020

[11] CryptoKitties, "*Getting Started*", https://guide.cryptokitties.co/guide/getting-started, 2017

[12] W. Radomski, A. Cooke, P. Castonguay, J. Therien, E. Binet, R. Sandford, "*ERC-1155: Multi Token Standard*", https://github.com/ethereum/EIPs/issues/1155, 2018

[13] G. Wood, "*Polkadot: Vision for a heterogeneous multi-chain framework*", https://polkadot.network/PolkaDotPaper.pdf, 2016

[14] S. Lee, "*ERC-809: Renting Standard for Rival, Non-Fungible Tokens*", https://github.com/ethereum/EIPs/issues/809, 2017

[15] Z. Yu, "*ERC-1201: Two Tiered Token Structure for Non-fungible Asset Ownership and Rental Rights*", https://github.com/ethereum/EIPs/issues/1201, 2018

***