Design of Extensible Non-Fungible Token Model in Hyperledger Fabric

Sangwon Hong*
Dept. of CSE, POSTECH
South Korea
sangwonhong@postech.ac.kr

Yoongdoo Noh*
Dept. of CSE, POSTECH
South Korea
yoongdoo0819@postech.ac.kr

Chanik Park
Dept. of CSE, POSTECH
South Korea
cipark@postech.ac.kr

ABSTRACT

This paper presents an extensible non-fungible token (NFT) model for supporting NFTs in Hyperledger Fabric (Fabric) with reference to ERC-721 defined as Ethereum standard NFT and Cosmos NFT. In this model, we defined the standard structure and interface for all Fabric NFTs. This model also supports the extensible structure and interface that accommodates different types of NFTs. To demonstrate how to take advantage of this model, we applied extensible NFTs such as document and signature tokens to a decentralized signature service.

CCS CONCEPTS

• **Software and its engineering → Software organization and properties → Software system structures → Software system models;**

KEYWORDS

blockchain, smart contract, non-fungible token, ERC-721, Cosmos NFT

ACM Reference format:

Sangwon Hong, Yoongdoo Noh, and Chanik Park. 2019. Design of Extensible Non-Fungible Token Model in Hyperledger Fabric. In *3rd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL '19), December 9–13, 2019, Davis, CA, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3366611.3368142

1 INTRODUCTION

Blockchains manage an append-only list of immutable ledgers, and runs programs called smart contracts on peer-to-peer systems in which every peer synchronizes the same states with other peers. Blockchains are divided into two types: permissionless and permissioned. Any participants can join in permissionless blockchains, e.g. Ethereum [1], regardless of any conditions; only participants registered in the membership list can join in permissioned blockchains e.g. Hyperledger Fabric (Fabric) [2]. Both Ethereum and Fabric manage decentralized applications (dApps) running on smart contracts.

Smart contracts can manage tokens that represent digital assets in blockchains. Tokens are divided into two types: fungible and non-fungible. Fungible tokens (FTs) are divisible into smaller units; non-fungible tokens (NFTs) are indivisible. Ethereum has both standard FT ERC-20 [3] and standard NFT ERC-721 [4] for interoperability between smart contracts. Fabric v2.0 Alpha proposed FabToken which supports FT, but does not support NFT yet even though NFT support in FabToken has been under discussion.

If dApps running on Fabric manage their digital assets, the assets must be digitalized with NFTs to make the assets unique and identifiable online. So Fabric needs to support NFT. Cosmos [5], a blockchain network connecting independent blockchains through inter-blockchain communication protocol, defined its own standard NFT Module [6] to transfer NFTs across multiple blockchains. In this paper, inspired by ERC-721 and Cosmos NFT, we designed Extensible NFT Model for Fabric (Figure 1). In our model, Fabric NFT has a standard structure and interface for interoperability between participants, and has an extensible structure and interface to meet various requirements of dApps. We prototyped a decentralized signature service to illustrate how our model can be applied.

2 SYSTEM MODEL

Note that if tokens come out from now on, they are regarded as NFTs. According to openzeppelin-contracts [7], ERC-721 has mapping tables for respectively managing token owners, token approvals, the counter for owned tokens, and operator approvals. ERC-721 does not manage token objects itself. But Cosmos NFT Module has a standard data structure for tokens. This module manages token objects. We defined a token as a key-value pair of

*Equal contribution

SERIAL '19, December 9–13, 2019, Davis, CA, USA

ACM ISBN 978-1-4503-7029-5/19/12…$15.00
https://doi.org/10.1145/3366611.3368142

CouchDB [8] in Fabric. The key is the token id. The value is a JSON that contains attributes for the token. Because of our definition of tokens in Fabric, our model has a data structure to manage token objects like Cosmos NFT Module. By adding attributes for ERC-721, our model can also support ERC-721.

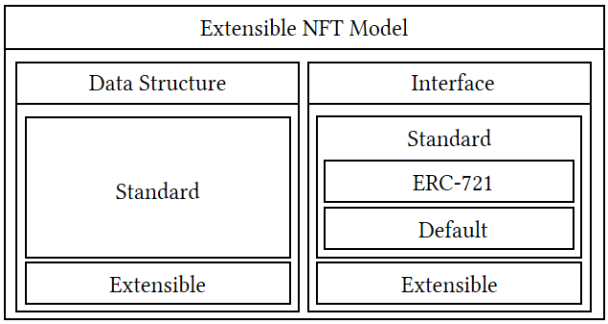

Figure 1: Overview of Extensible NFT Model for Fabric

The data structure of our model consists of standard and extensible structures. In the standard structure, attributes *owner*, *operator*, and *approvee* referenced from ERC-721 represent the user's role in the token. Attribute *owner* is a user who has ownership of his token. Attribute *operator* is a user who manages all tokens owned by the *owner*. Attribute *approvee* is a user with permission to transfer the token to others. Depending on dApps, extensible attributes *xattr* (extended attributes) and *uri* (uniform resource identifier) can support various types of tokens by managing the extensible structure containing sub-attributes. Attribute *xattr* manages sub-attributes stored on on-chain. The sub-attributes for *xattr* are implemented by developers who define their own token types. Attribute *uri* manages sub-attributes *path* and *hash*; sub-attribute *path* is the path of off-chain storage that stores the metadata for the token; sub-attribute *hash* is the Merkle root created by the Merkle tree of which the leaves are data stored on off-chain storage.

The interface of our model consists of standard and extensible interfaces. The standard interface consists of ERC-721 and default interfaces. The ERC-721 interface was designed to provide the ERC-721-compatible specification. In the ERC-721 interface, function *balanceOf* counts all tokens owned by the *owner*. Function *ownerOf* returns the token *owner*. Function *transferFrom* changes the token *owner*. Functions *approve* and *getApproved* respectively set and retrieve the *approvee* for the token. Function *setApprovalForAll* adds or removes the token *opertator*, and function *isApprovedForAll* queries whether a user is the *operator* for the *owner*.

The default interface was designed to provide the standard specification for Fabric itself. In this interface, functions *mint* creates a token, and function *burn* removes a token. The other functions are getter and setter functions to retrieve and update all attributes. But attributes *id* and *type* do not have setter functions because they should not be updated after initialization with function *mint*.

To retrieve and update off-chain and on-chain extensible attributes, the extensible interface defines getter and setter functions for attributes *uri* and *xattr*.

3 USE CASE

We applied our model to a decentralized signature service in which users digitally sign electronic documents to ensure integrity through Fabric blockchain. This service manages document and signature tokens. In the document token, the hash of the document, the list of signers, and the list of signature token IDs for the signers are managed on on-chain through *xattr*. Document files are stored on off-chain storage managed by *uri*. In the signature token, the hash of the signature image is managed on on-chain through *xattr*. Signature images are stored on off-chain storage managed by *uri*. In conclusion, our model provides various types of NFTs with the standard as well as the extensible structure and interface using *xattr* and *uri*.

4 CONCLUSION

We designed Extensible NFT Model to support NFTs in Fabric with reference to ERC-721 and Cosmos NFT. Our model has standard and extensible components that consists of the data structure and the interface. To maximize benefits of our model, Fabric needs to support inter-channel communication protocol, then our model can allow participants to transfer their NFTs across channels.

ACKNOWLEDGMENTS

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Consilience Creative program (IITP-2019-2011-1-00783) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). We also appreciate the support from POSCO ICT.

REFERENCES

[1] V. Buterin, A Next Generation Smart Contract & Decentralized Application Platform, Ethereum Foundation, 2015.
[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis,A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. Weed Cocco, and J. Yellick, Hyperledger fabric: a distributed operating system for permissioned blockchains, In *EuroSys '18*, 2018.
[3] ERC-20, https://eips.ethereum.org/EIPS/eip-20.
[4] ERC-721, https://eips.ethereum.org/EIPS/eip-721.
[5] J. Kwon, and E. Buchman, A Network of Distributed Ledgers, Cosmos, 2018.
[6] Cosmos NFT Module, https://hackmd.io/@okwme/cosmos-nft.
[7] OpenZeppelin, https://github.com/OpenZeppelin/openzeppelin-contracts.
[8] CouchDB, https://couchdb.apache.org/.