OpenDSU & Tokenisation

Report: OpenDSU & Tokenisation
Author: Sînică Alboaie, Axiologic Research
Version: 1.0.
Date: December 2023

Abstract

Data Sharing Units (DSUs), a fundamental concept of OpenDSU, can be utilised to implement non-fungible tokens (NFTs). For fungible tokens, smart contracts that require global consensus for all data at the level of the distributed ledger are the appropriate solution. One advantage of DSUs over other methods of implementing NFTs is that they store unique content, metadata, and properties associated with NFTs off-chain in a standardised and cryptographically secure manner, unlike other storage methods that rely on HTTP-accessible systems. The NFT's associated content and all its versions stored as a DSU are, therefore, automatically verifiable.

Anchoring a DSU in a ledger ensures 'freshness' guarantees, confirming that we have the latest version. Any other cryptographic properties related to immutability are automatically verified through anchoring and the way DSUs are stored in immutable, content-addressable “bricks”. An anchor of a DSU is a nano-ledger, which is a cryptographic sequence of append operations for a new version, creating opportunities for storing anchors in distributed ledgers that ensure content replication and 'freshness' without the need for global consensus on all objects stored in the ledger, offering the potential for creating new types of lighter and more suitable ledgers for complex enterprise blockchain solutions that require high scalability.

From a confidentiality and privacy perspective, DSUs offer a higher level of protection by encrypting all content, which can only be decrypted by sharing reading keys derived from a typically secret unique 'seed' key held by the current DSU owner. As will be seen in the current document, anchoring also provides a simple mechanism for changing ownership by allowing the appending of a new version with a new owner, i.e., a new 'seed' key that will be used in the future to validate new 'append version' operations for the DSU.

In this document, we will provide a sufficiently detailed explanation of how anchors function and the process involved in changing ownership. This document will not delve into implementation specifics, as these concepts are general enough to allow for implementation across various blockchain technologies. Our experience implementing OpenDSU using Ethereum smart contracts and Hyperledger Fabric chain code illustrates this versatility.

Anchors as Nano-Ledgers. Ownership Change

Understanding the concept of DSUs fundamentally involves the idea that the off-chain content anchoring or notarisation of a DSU is done in the form of anchors, which are data structures stored in a distributed ledger, typically managed by a smart contract.

An anchor contains an identifier and a cryptographically chained list of the DSU's versions. The anchor's identifier, often called AnchorID, includes a public key to validate whether a new version can be appended to the existing list. Accepting a new version involves verifying the current owner's signature; in this sense, information from the AnchorID is used. AnchorID also includes information related to the BDNS (Blockchain Domain Naming System), allowing for data segregation by organisations, departments, applications, and use cases. However, we will not delve into these aspects in this document. It suffices to mention that different entities can use the same anchoring ledger for storing bricks (the off-chain content of the DSUs).

AnchorIDs and the base keys (seeds) used to derive encryption keys and the private key associated with the public key exposed in an AnchorID are represented as SeedSSI (SSI stands for Self-Sovereign Identifiers) for uniformity and to enable this segregation across domains facilitated by the BDNS. Additionally, even the versions of the DSU are referred to as KeySSIs because a new version can be stored by a different organisation (and, therefore, a different BDNS domain) than the original organisation. This structure allows for flexible management of off-chain storage across various organisational boundaries.

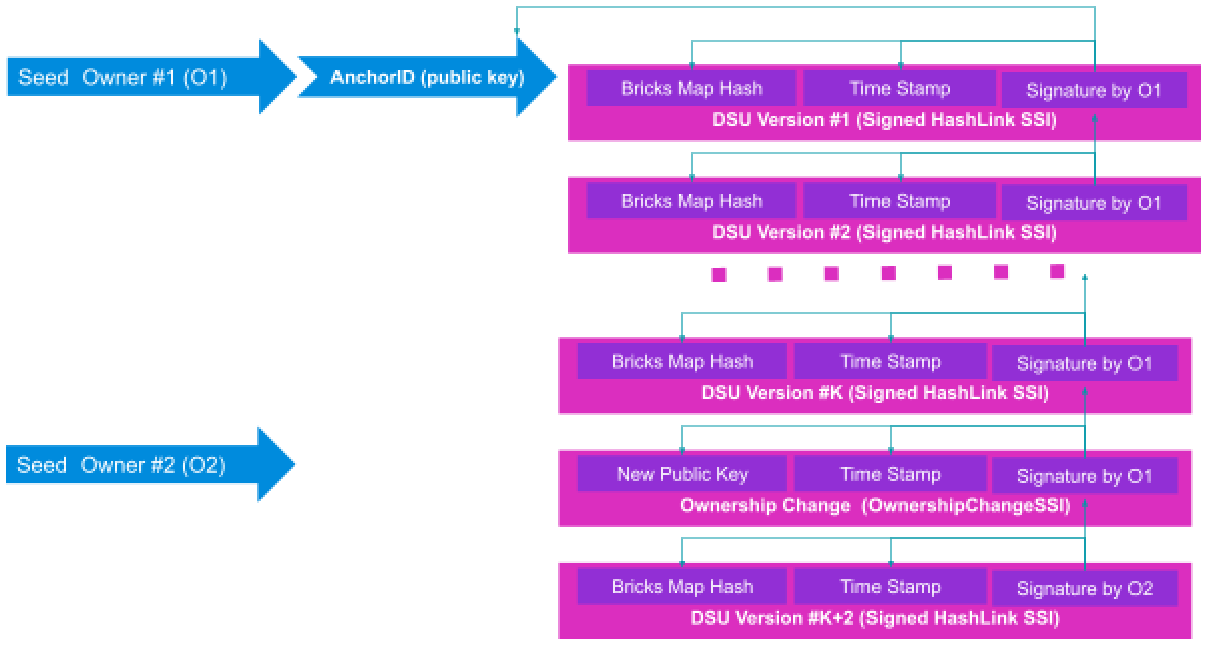

Diagram 1: Anchors as nano-ledgers. Ownership Change.

In this document, we will focus on explaining how NFTs based on DSUs function and will not delve into other details. For a deeper understanding of DSUs, anchoring, and KeySSI concepts, we recommend the films and documents available on the OpenDSU.org website.

Each KeySSI, which represents a new version, includes mandatory BDNS domain information, which is present in all KeySSIs, as well as the hash link of the BrickMap for that version and a timestamp marking when that version was created. Additionally, this information, along with the AnchorID and the list of previous versions in the anchor if present, is signed with the private key of the current owner, a key derived from the seed that must be kept as secure as possible by the owner of the DSU (or the NFT if we are referring to NFTs).

At this point, it's important to notice why we refer to anchors as nano-ledgers. Each anchor functions like a “minimalistic smart contract” with only one operation: appending a new version. Each version is effectively a transaction signed by the current owner. This design ensures that each update or addition to the DSU is securely recorded, providing a clear and tamper-evident history of changes without relying on the ledger used for anchoring.

The sole role of the ledger used for anchoring is to ensure the "freshness" property of the anchor, meaning to confirm that when it is used, we have the latest version. Other typical properties blockchain technology offers are covered by how anchors are structured and chained through digital signatures. It's as if the nano-ledger functions like a blockchain with chained blocks having a single transaction each. This structure reduces the blockchain role while providing essential security and verification features typical of complex blockchain environments.

In OpenDSU terminology, we speak of micro-ledgers as ledgers that contain only one smart contract and refer to them as SVDs (Self Validating Data). These SVDs can be stored within DSUs and offer an exciting alternative to standard “smart contracts” because they can be made available to a limited number of stakeholders, becoming what we sometimes call a "secret" smart contract in our articles. The nano-ledger represented by the anchor has a single operation, and a minimal number of transaction types are possible. In contrast, micro-ledgers, represented as SVDs, allow unlimited operation types and transactions and are programmable in limitless ways. For more details about SVDs and how they might be used, we recommend RFC-036, which is available on the OpenDSU website.

A simple expansion of the types of operations allowed by nano-ledgers is to permit, in addition to the operation or transaction of adding a new version, the change of ownership of the DSU, i.e., changing the public key that allows the addition of new versions. Of course, the AnchorID cannot be changed, and thus, the identity of the DSU will remain tied to the initial "seed" used for creation. However, after the change of ownership, the only allowed signatures will be made with a new private key generated from the "seed" of the new owner.

The only modification is that instead of adding a version that contains a hash link to a version brick, we put in a special KeySSI named OwnershipChangeSSI, which includes the new public key. The old owner will sign this pseudo-version, but by convention, the smart contract for anchoring will now allow the appending of future anchors or ownership changes only if the future modifications are signed and match the new public key. This mechanism ensures that while the foundational identity of the DSU remains constant, the control can securely transition to a new owner, reflecting updates in ownership clearly and securely in the blockchain ledger.

Typical NFT Operations as operations on DSUs

This chapter starts with a small table that overviews typical operations for managing Non-Fungible Tokens (NFTs). It details critical functions such as creating new tokens (minting), transferring ownership, querying current ownership, permanently removing tokens (burning), updating associated metadata, and managing permissions for delegation. Each operation is crucial for effectively administrating and utilising NFTs, ensuring secure, transparent, and efficient handling of these digital assets on the blockchain. This introduction sets the stage for understanding how NFTs can be implemented using the DSU framework, highlighting the essential operations needed for their lifecycle management.

NFT Operation	**Explanation**
Mint	Create a new unique token, typically adding details about the asset it represents.
Ownership Transfer	Transfer ownership of an NFT from one account to another.
Owner Inquiry	Query the current owner of a specific NFT.
Burn NFT	Remove an NFT permanently, often used when an asset needs to be retired or is no longer valid.
Metadata Update	Update metadata associated with an NFT, such as digital files, descriptions, or attributes that define the asset's characteristics.
Delegation	Manage who can operate or transfer the NFT on behalf of the owner.

Table 1: Typical NFT operations

The following table summarises the corresponding actions within the Data Sharing Unit (DSU) framework for each typical Non-Fungible Token (NFT) operation. To mint a new NFT, a new DSU is created along with a random "seed" key, which is important for controlling the appending of new versions to the DSU. For ownership transfer, the DSU undergoes an update by appending a new version controlled by a new key, a process outlined in the previous discussion. Inquiring about the owner involves checking the current public key that controls the DSU, ensuring transparency in ownership.

To effectively burn an NFT, a version of the DSU is appended that either has no control public key or possesses an invalid one, and, if appropriate, the associated storage bricks can be deleted to finalise the removal. Metadata updates are facilitated by appending a new version of the DSU with revised metadata, allowing for the dynamic updating of information associated with the NFT. Lastly, the delegation process involves transferring control of the DSU by passing the "seed" key to an intermediary party, enabling them to manage the DSU on behalf of the original owner. Additionally, other cryptographic-based delegations can be conceptualised and implemented at the anchoring level to extend the functionality and flexibility of DSU operations.

These operations provide a robust framework for managing NFTs within the blockchain, ensuring secure and efficient handling of digital assets.

NFT Operation	**DSU Corresponding actions**
Mint	Create a new DSU and a new random “seed” key to control the appending of new versions of the DSU.
Ownership Transfer	Transfer ownership of the DSU by appending a version that is controlled by a new key (explained in the previous chapter)
Owner Inquiry	Check the current public key that currently controls the DSU.
Burn NFT	This can be achieved by appending a DSU version that has no control public key or an invalid public key. Additionally, the equivalent bricks could be deleted.
Metadata Update	This can be achieved by appending a new DSUSu version with updated metadata.
Delegation	This operation requires the transfer of the DSU “seed” to the intermediary party to which it is delegated to control the DSU on behalf of the owner. If necessary, other cryptography-based delegations can be imagined and implemented at the anchoring level.

Table 2:Coresponding DSU operations

ZKP & Wallets Considerations

Implementing NFTs with OpenDSU nearly eliminates the need for Zero-Knowledge Proofs (ZKP) for tokenisation, as access to data and metadata associated with NFTs is stored in DSUs designed with privacy and confidentiality in mind. However, the conversation shifts when discussing fungible tokens. For more sophisticated tokens that allow simultaneous state changes, ZKP techniques can still be utilised, but ideally, ZKP proofs for ensuring computational integrity should be stored within DSUs. This approach minimises the attack surface in case of vulnerabilities in ZKP techniques.

OpenDSU employs robust cryptographic primitives such as hashes, symmetric encryption, and digital signatures, significantly reducing the likelihood of cryptographic vulnerabilities compared to other systems. Furthermore, to implement "computational integrity" constraints in smart contract instances at the NFT level, techniques such as SVDs and mechanisms for deriving and managing keys using KeySSIs and other OpenDSU-provided mechanisms can be used. Generally, all blockchains require some form of key management, and it is noteworthy that DSUs themselves act as secure containers for private data, making them well-suited for implementing wallets designed to manage private or secret keys. This alignment makes OpenDSU a safe and efficient framework for managing NFTs and other blockchain-based assets.

Conclusions

The implementation of tokenisation using OpenDSU introduces a sophisticated framework that integrates Data Sharing Units (DSUs) to manage non-fungible tokens (NFTs) effectively. This approach leverages DSUs to store off-chain content, metadata, and properties associated with NFTs in a standardised and cryptographically secure manner.

One significant advantage of using DSUs is enhancing the blockchain's flexibility in choosing underlying on-chain technologies. The DSU structure, facilitated by advanced KeySSI mechanisms, allows for the sophisticated management and derivation of keys, ensuring robust data segregation and security. This framework supports the evolution of distributed ledger technologies beyond their initial cryptocurrency applications, paving the way for robust enterprise blockchain platforms that require high scalability and enhanced security features.

Furthermore, DSUs provide high confidentiality and privacy by encrypting all DSU’s content and making it accessible only through keys derived from a securely held 'seed.' This system maintains privacy and ensures data integrity and security, which are paramount in enterprise applications.

The anchoring mechanism within the DSU framework uses nano-ledgers or anchors—a form of minimalistic “smart contracts” —that contain a cryptographically chained list of the DSU's versions. This anchor system supports operations such as appending new versions and ownership transfers without necessitating a global consensus, significantly reducing the latency and resource consumption typically associated with traditional blockchain transactions.

Moreover, the DSU's flexibility is highlighted by its ability to support micro-ledgers within the DSUs, described as SVDs (Self Validating Data). These can act as “private smart contracts” accessible to a limited number of stakeholders, offering an alternative to standard smart contracts' public and pervasive nature. It also provides a method of achieving the same goals as ZKP using only standard cryptography. This aspect is particularly beneficial for enterprises that require simplicity and process audibility with the best confidentiality and data integrity.

In conclusion, OpenDSU's approach to using DSUs and KeySSIs provides a flexible, secure, and efficient blockchain tokenisation framework. This method enhances scalability and performance and introduces advanced key management and privacy features critical for modern blockchain implementations in enterprise settings. DSUs' inherent capabilities to adapt and evolve with changing technology landscapes make them a potent tool for the future development of blockchain technology, moving beyond cryptocurrencies-inspired architecture to support a more comprehensive array of business applications.