ASOBI MARKET Architecture

Justus Perlwitz, Asobimo, Inc.

2018-07-04 v4

Abstract

In this document we will explain some of the architecture components of the ASOBI MARKET. The architecture detailed in this document are patent pending under PCT/JP2018/25247.

Contents

1 Changelist **1**
- 1.1 2018-07-04 (v4) . . . 1
  - 1.1.1 Added . . . 1
- 1.2 2018-07-03 (v3) . . . 1
  - 1.2.1 Added . . . 1
  - 1.2.2 Fixed . . . 1
- 1.3 2018-07-02 (v2) . . . 1
  - 1.3.1 Added . . . 1
  - 1.3.2 Changed . . . 2
  - 1.3.3 Fixed . . . 2

2 Glossary **2**

3 Architecture Overview **3**

4 Smart Contracts **5**
- 4.1 ASOBI COIN . . . 5
- 4.2 License Storage . . . 5
- 4.3 ASOBI MARKET . . . 5

5 ASOBI COIN **6**
- 5.1 Purchasing ASOBI COIN on DEX (On-Chain) . . . 6
- 5.2 Purchasing ASOBI COIN on a Centralized Exchange (CE) . . . 6
- 5.3 Transferring ASOBI COIN among peers . . . 6
- 5.4 Generic Interaction between ASOBI COIN and other smart contracts . . . 8

6 Content Management **8**
- 6.1 Adding Content . . . 8
- 6.2 Removing Content . . . 8

7 Content Exchange **8**
- 7.1 Content Purchase (from ASOBI MARKET) . . . 8
- 7.2 Content Exchange (Mediated) . . . 10
- 7.3 Content Exchange (Peer-To-Peer) . . . 11

8 Viewer Application **11**
8.1 Signing View Requests . . . 11
8.2 Requesting Decryption Keys . . . 12
8.3 Requesting Decryption Keys (Without License) . . . 12
8.4 Viewing Content . . . 14

9 Viewer DRM **14**

10 Governance and Smart Contract Ownership **14**
10.1 Ownership . . . 15
10.2 Voting on Proposals . . . 15

11 Appendix A. References **15**

12 Appendix B. List of Figures and Tables **18**

1 Changelist

Explanation of following change items: `-` indicates that a line was removed, `+` indicates that a line was added.

1.1 2018-07-04 (v4)

1.1.1 Added

- Short abstract with patent number

1.2 2018-07-03 (v3)

1.2.1 Added

- Figure 9: Show the flow of ASOBI COIN from *User B* to all receivers during mediated content exchange.

1.2.2 Fixed

- Section 7.2.: The correct amount *User A* will receive now is $M = N - R - S$.

1.3 2018-07-02 (v2)

1.3.1 Added

- Numbered all sections.
- Explained ASOBIMO abbreviation.

1.3.2 Changed

- Subsection 7.1.: License Storage -> ASOBI MARKET

```
-We also note that the License Storage contains all the business logic required
+We also note that the ASOBI MARKET contains all the business logic required
```

- Subsection 7.2.: Use ASOBI MARKET for content exchanges instead of the license storage
- Subsection 8.4.: The viewer DRM now receives the decryption keys directly.
- Figure 5: Simplify delegated contract interaction.
- Figure 8: ASOBI MARKET now shares the publisher and ASOBIMO revenue directly.

```
-    ABM => "License Storage" [label="Transfer\nR+S ABX"];
+    ABM -> ASOBIMO [label="Transfer R"];
+    ABM -> Publisher [label="Transfer S"];
     ABM => "License Storage" [label="Create\nLicense X\nfor Y"];
     "License Storage" => "License Storage" [label="Store\nLicense ID\nfor X"];
     "License Storage" => "License Storage" [label="Assign\nLicense\nto User"];
-    "License Storage" -> Publisher [label="Transfer S"];
-    "License Storage" -> ASOBIMO [label="Transfer R"];
```

- Figure 9: Use ASOBI MARKET in the mediated content exchange.
- Figure 13, 14: The Viewer DRM now received the decryption keys directly.

1.3.3 Fixed

- Fix typographic and stylistic mistakes
- Subsection 4.2.: License ID should have been 3, not 2.

```
 | User ID  | License ID   |
 |---------:|--------------|
 | 1        | 1, 2, \ldots |
-| 2        | 2 $\ddots$   |
+| 2        | 3 $\ddots$   |
 | $\vdots$ | $\vdots$     |
```

- Subsection 5.4.: Fixed the figure reference for figure 5.
- Figure 3: Users pay M (JPY) to purchase N (ABX). Before it was the other way around.

```
-    User => Exchange [label="Buy M"];
-    User => CC [label="Authorize spending N"];
+    User => Exchange [label="Buy N"];
+    User => CC [label="Authorize spending M"];
```

- Figure 5: C spends N on behalf of the users and ASOBI COIN calls F.

```
-    User => "ASOBI COIN" [label="Authorize C to spend N\nand call N"];
+    User => "ASOBI COIN" [label="Authorize C to spend N\nand call F"];
```

- Figure 18: Clarify that the proposal expired even though it was approved.

2 Glossary

Table 1: Glossary

Name	Definition
ABM	ASOBI MARKET
ABX	ASOBI COIN
ASOBI COIN	ERC-20 based virtual currency
ASOBI MARKET	Decentralized Market Place
ASOBIMO	Asobimo, Inc.

Name	Definition
Blockchain	Transaction storage that uses cryptographic methods to ensure integrity between transactions
CE	Centralized Exchange
CID	Unique Content ID
CMS	Content Management System
DEX	Decentralized Exchange
DRM	Digital Rights Management
ERC-20	Standard for creating virtual currencies based on Ethereum blockchain
Ethereum	Cryptocurrency and Distributed Smart Contract platform
IPFS	InterPlanetary File System: Open and Decentralized Peer to Peer file storage
MSW	Multi-Signature Wallet
Pinning	Ensuring file content associated with CID stays on IPFS

3 Architecture Overview

We refer to fig. 1 for an overview of the architecture. Marked in rectangles are data stores. Decentralized computer programs are marked in ovals. Centralized computer programs are marked in octagons. A brief explanation of every architecture element follows in tbl. 2.

Table 2: Architecture Term Explanation

Architecture Element	Explanation
ASOBI COIN	Smart Contract. Store ASOBI COIN balances.
Book Content	IPFS storage for encrypted e-books.
Content Management System	Manages publisher content.
DRM Server	Centralized service that manages encryption and decryption of content.
Ethereum Blockchain	Stores and executes smart contracts.
IPFS	Stores encrypted publisher content.
Key Storage	Securely store content encryption and decryption keys.
License Storage	Smart Contract. Store which licenses users own.
Music Content	IPFS storage for encrypted music content.
Shop	Allows users to buy content from the ASOBI MARKET.
Trading Platform	Allows users to exchange content licenses among each other.
Video Content	IPFS storage for encrypted video content.
Viewer DRM	Trusted content decryption core that users cannot access.
Viewer	Allows users to view content for which they own a license.

4 Smart Contracts

4.1 ASOBI COIN

The ASOBI COIN smart contract stores the ASOBI COIN balance for every user as in tbl. 3 . One should keep in mind that the smart contract never stores a complete state of the balances table, but only stores the differences – as transactions.

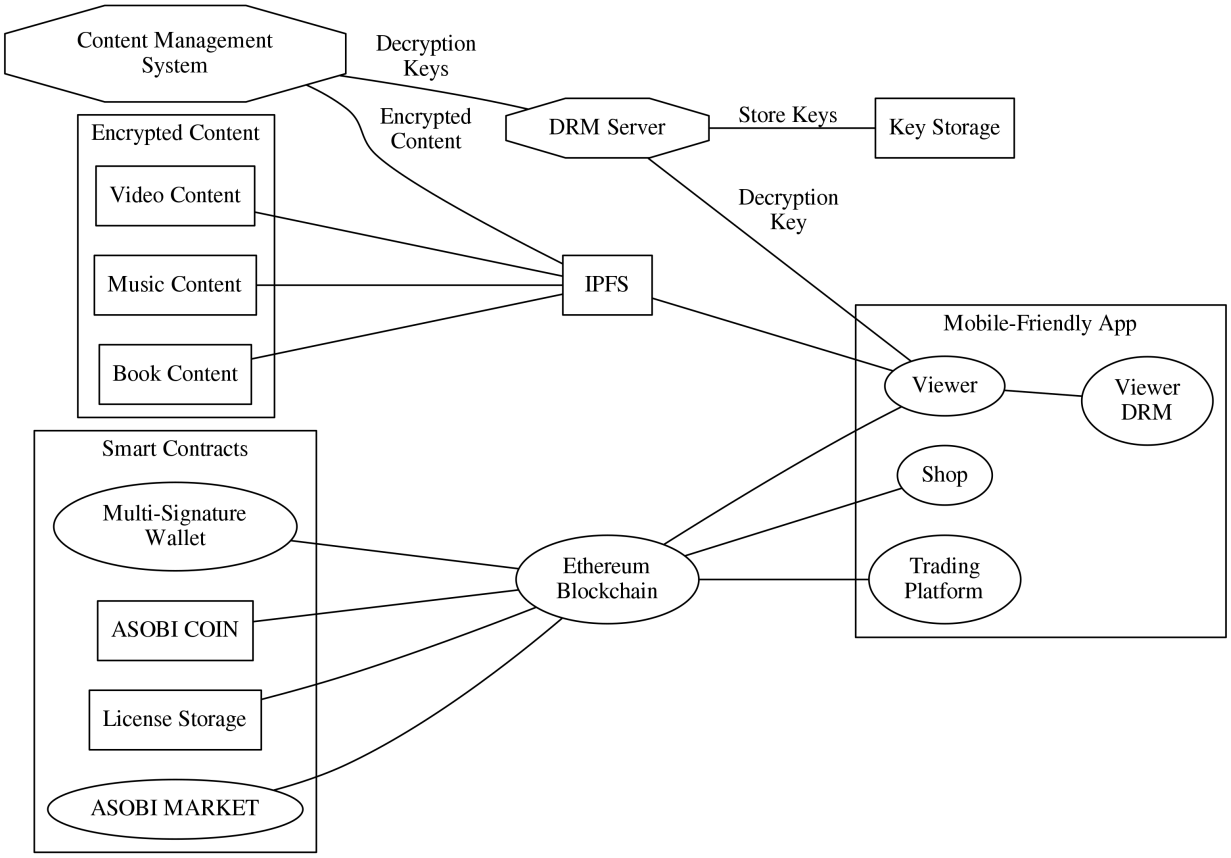

Figure 1: Architecture Overview

Table 3: ASOBI COIN: User Balances

User ID	Balance
1	1003
2	98
⋮	⋮

4.2 License Storage

The License storage keeps record of which licenses exist as in tbl. 4, and which user these licenses belong to. A content ID is assigned to every license like in tbl. 5, and a user is never associated directly with the content.

Table 4: License Storage: User Licenses

User ID	License ID
1	1, 2, …
2	3 ⋱
⋮	⋮

Table 5: License Storage: Content Belonging to License

License ID	Content ID
1	93
2	46
3	1342

4.3 ASOBI MARKET

The ASOBI MARKET can create new licenses in the license storage. It keeps a record of all the content it can create licenses for and a unique publisher-specific identifier. Furthermore, it stores revenue-share properties for every content ID. This is further detailed in tbl. 6.

Table 6: Content Properties

Content ID	Publisher ID	Revenue Share ASOBIMO	Revenue Share Publisher
46	`ROMEO AN..`	93	7
93	`GRUNDLAGEN`	90	10
1342	`MANIFEST D`	50	50
⋮	⋮	⋮	⋮

5 ASOBI COIN

5.1 Purchasing ASOBI COIN on DEX (On-Chain)

In this scenario, a user buys N ASOBI COIN using another token on the same blockchain using a blockchain transaction. We call the token *TOKEN* and the agreed upon purchase price in *TOKEN* exactly M tokens. The user purchases the token through a decentralized exchange contract *DEX*. We assume that *DEX* is offering to sell the user M ASOBI COIN for N token. See fig. 2 for more details.

We note that the user authorizes the *TOKEN* contract to spend M on their behalf in order to purchase N ASOBI COIN. This way, we can have the whole purchase take place in one blockchain transaction. The alternative would be to have two transactions:

1. User asks *TOKEN* to authorize *DEX* to spend M.
2. User asks *DEX* to purchase N ASOBI COIN, and the *DEX* will transfer the required amount itself.

We use *TOKEN* as a sort of proxy to orchestrate the whole transaction.

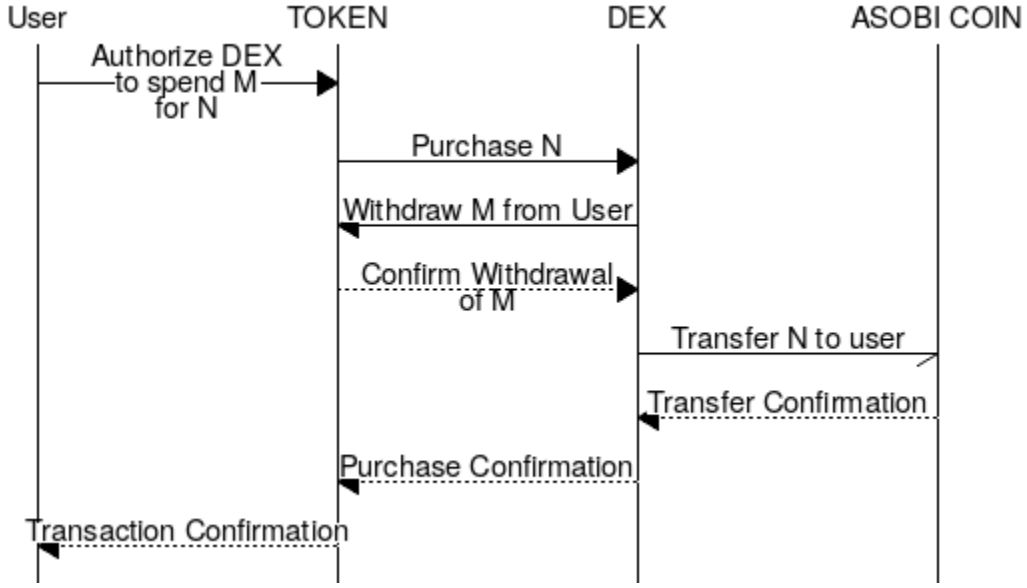

Figure 2: Purchasing ASOBI COIN on DEX (Single Transaction)

5.2 Purchasing ASOBI COIN on a Centralized Exchange (CE)

In this scenario, a user buys N ASOBI COIN using a fiat currency on a centralized exchange. The currency in this example is Japanese Yen (JPY). Let M be the purchase price in JPY for N ASOBI COIN. Let *CC* be the payment provider the user uses to purchase N ASOBI COIN. We refer to fig. 3 to understand the sequence of events. Only the interaction between the Exchange and ASOBI COIN happen in one atomic blockchain transaction.

5.3 Transferring ASOBI COIN among peers

In this example there are two users, *User A* and *User B*. *User A* wants to transfer N ASOBI COIN to *User B*. We refer to fig. 4 to understand the sequence of events. We also assume that *User A* has enough funds for the transfer.

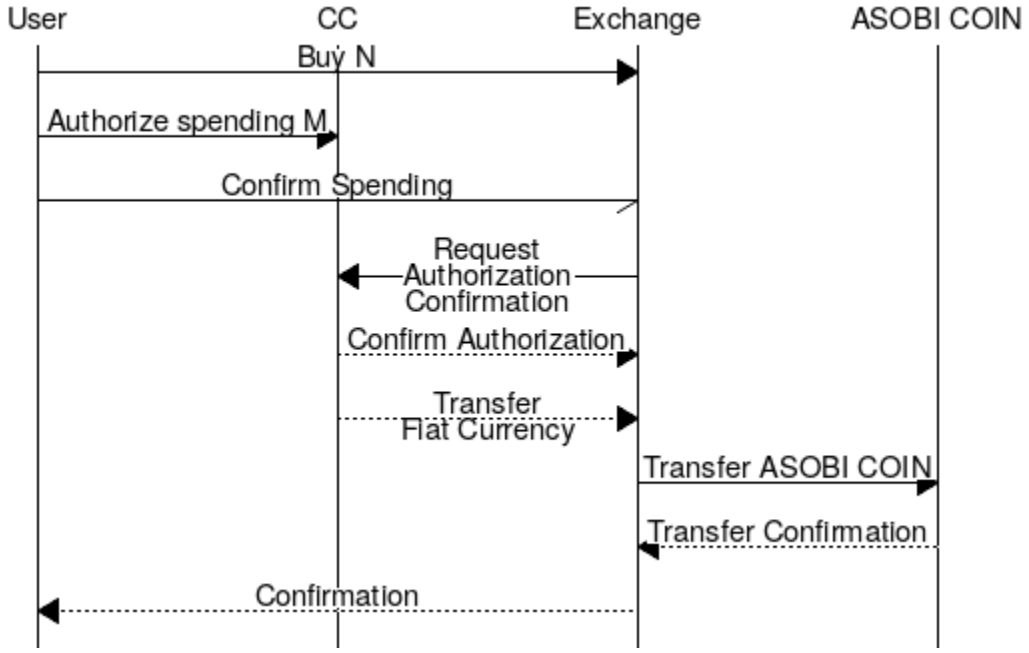

Figure 3: Purchasing ASOBI COIN on CE

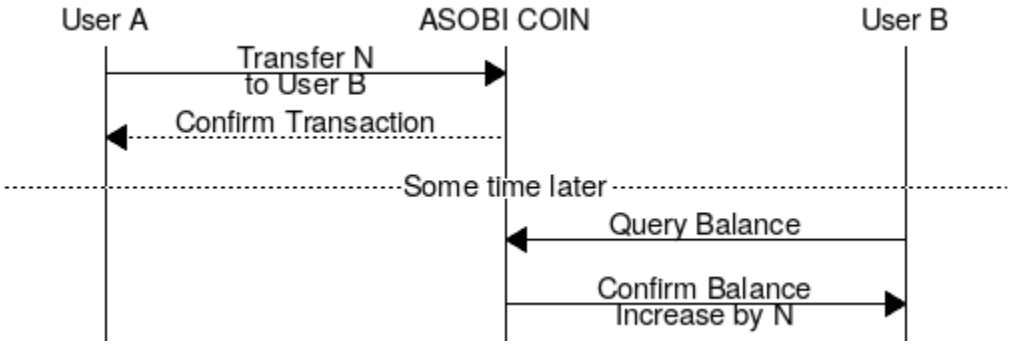

Figure 4: Transferring ASOBI COIN among peers

5.4 Generic Interaction between ASOBI COIN and other smart contracts

If we desire to reduce the amount of transactions, we can use ASOBI COIN directly to not only approve other contracts to spend a user defined amount, but we can also instruct it to call the other contract directly on behalf of the user. Common standards for implementing this type of behavior are ERC-223 and ERC-777. Let N be the amount of ASOBI COIN the user wants to spend and F the specific smart contract functionality that the user wants to call immediately from smart contract C. We refer to fig. 5 to understand the exact sequence of events in one transaction.

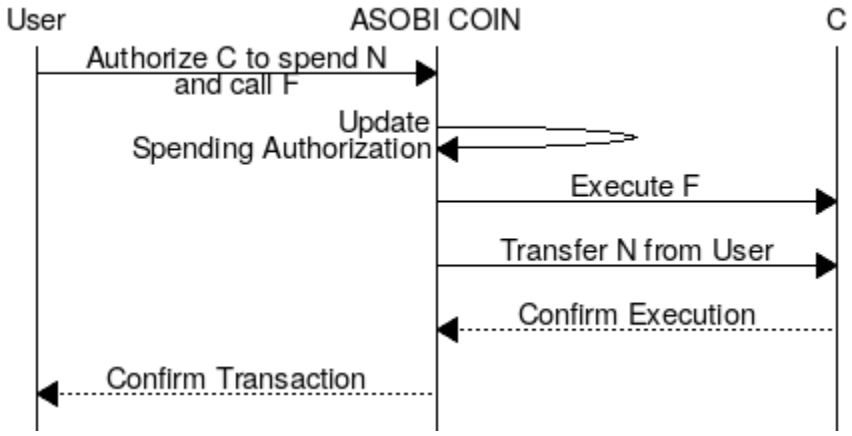

Figure 5: Generic ASOBI COIN interaction (single transaction)

6 Content Management

6.1 Adding Content

Adding Content is described in fig. 6.

6.2 Removing Content

Removing Content is described in fig. 7.

7 Content Exchange

7.1 Content Purchase (from ASOBI MARKET)

See fig. 8.

Here we use license X as a placeholder for any arbitrary content license for content Y that the user might acquire. We furthermore define the purchase price of X in ABX token to be P. Let R be the revenue share paid to *ASOBIMO*, let S be the revenue share paid to the *Publisher*.

We also note that the ASOBI MARKET contains all the business logic required to pay revenue shares to publishers and ASOBIMO.

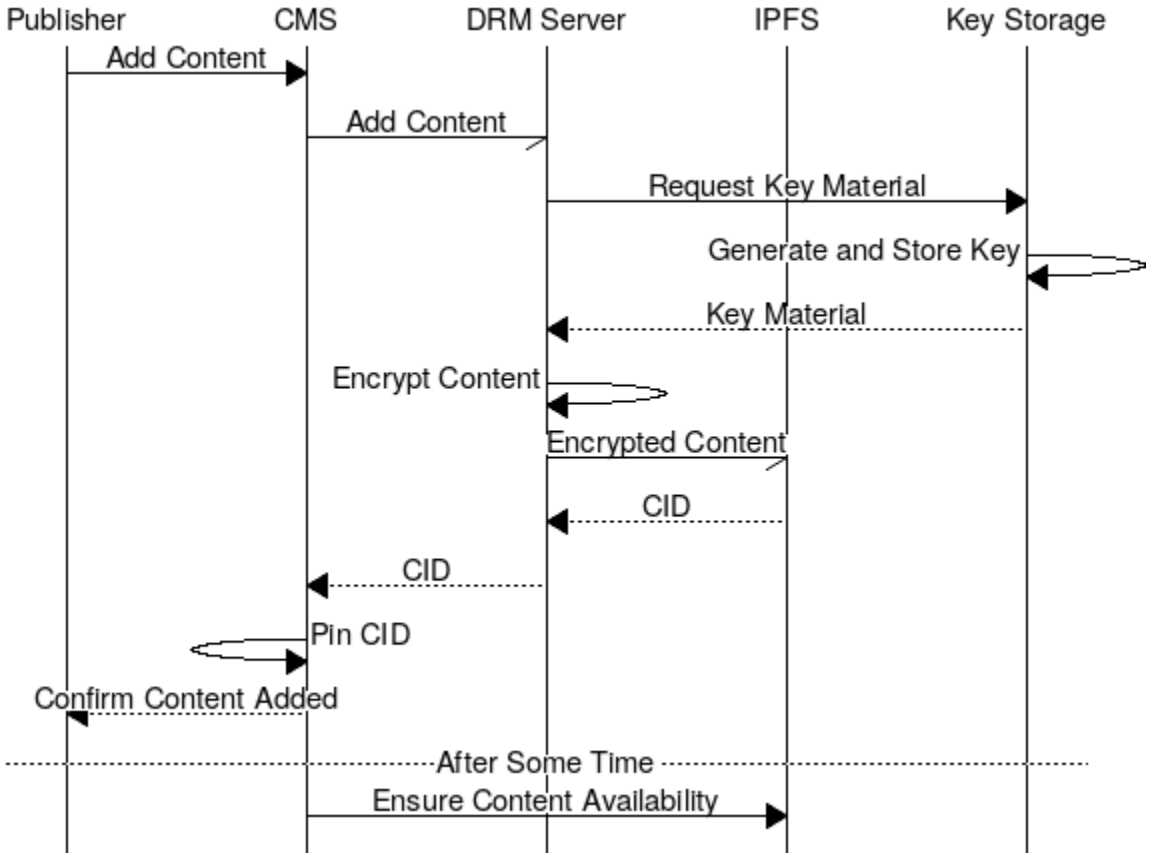

Figure 6: Adding Content to CMS

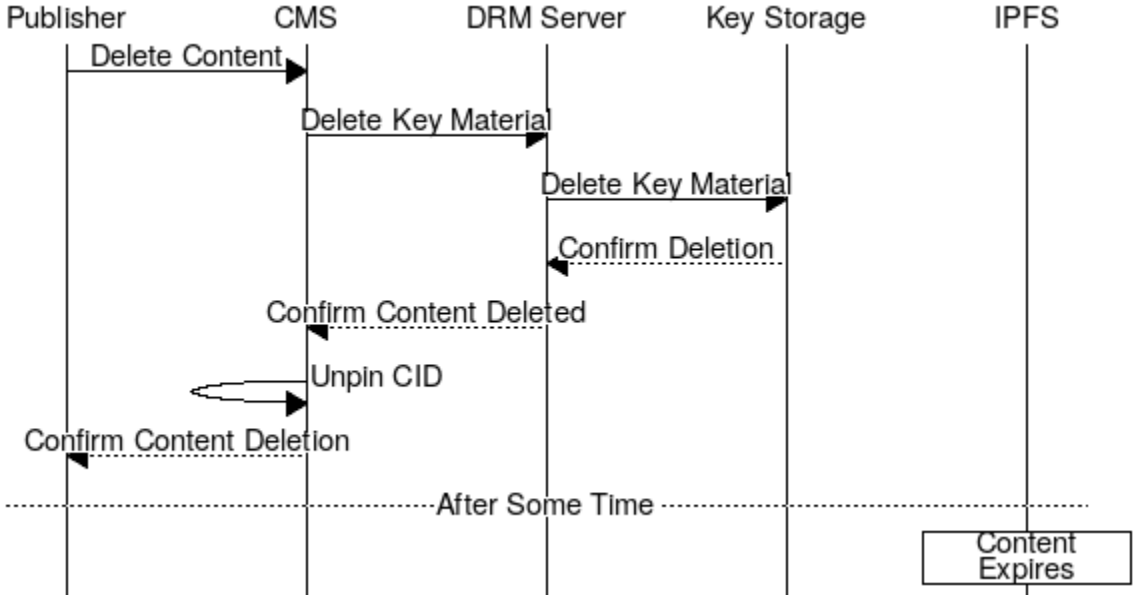

Figure 7: Removing Content from CMS

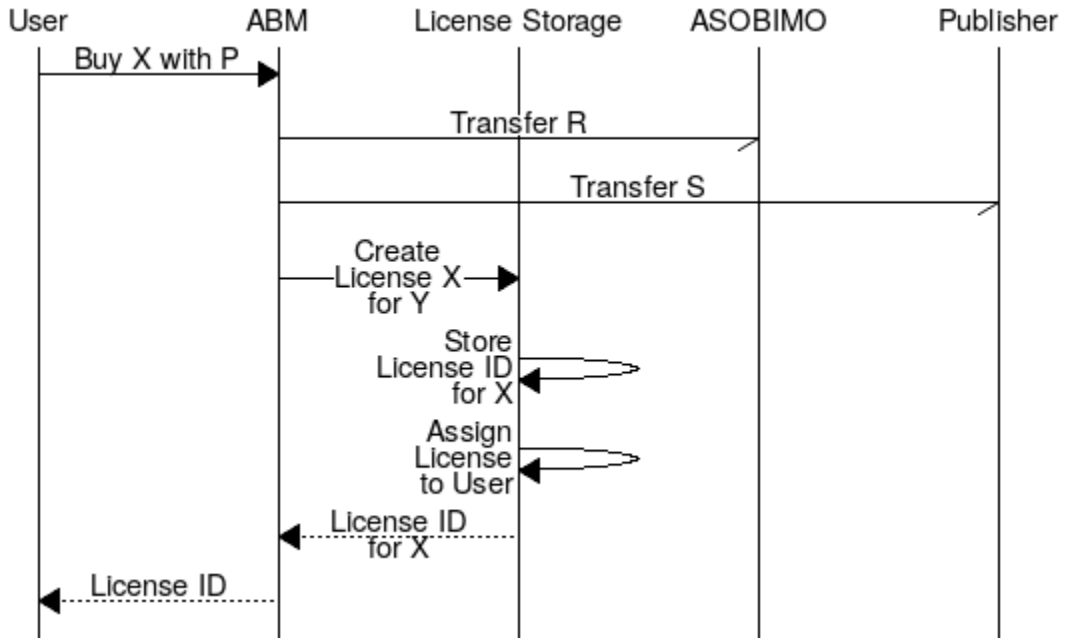

Figure 8: Content Purchase (Single Transaction)

7.2 Content Exchange (Mediated)

In this scenario, *User A* wants to sell their license *X* for content *Y* to any user for *N* ASOBI COIN on the ASOBI MARKET. We refer to fig. 10 to understand the exact sequence of events.

Let *R* be the revenue share paid to *ASOBIMO*, let *S* be the revenue share paid to the *Publisher*. The actual amount *User A* will receive is called *M* and we define it as *M=N-R-S*. The ASOBI MARKET functions as a trusted Escrow between *User A* and *User B* in this case and it

- atomically swaps a user license and ASOBI COIN
- contains the business logic for paying transaction revenue shares according to ASOBI MARKET rules

Both the buyer, as well as seller will interact through that contract.

The resulting flow of ASOBI COIN is illustrated in fig. 9.

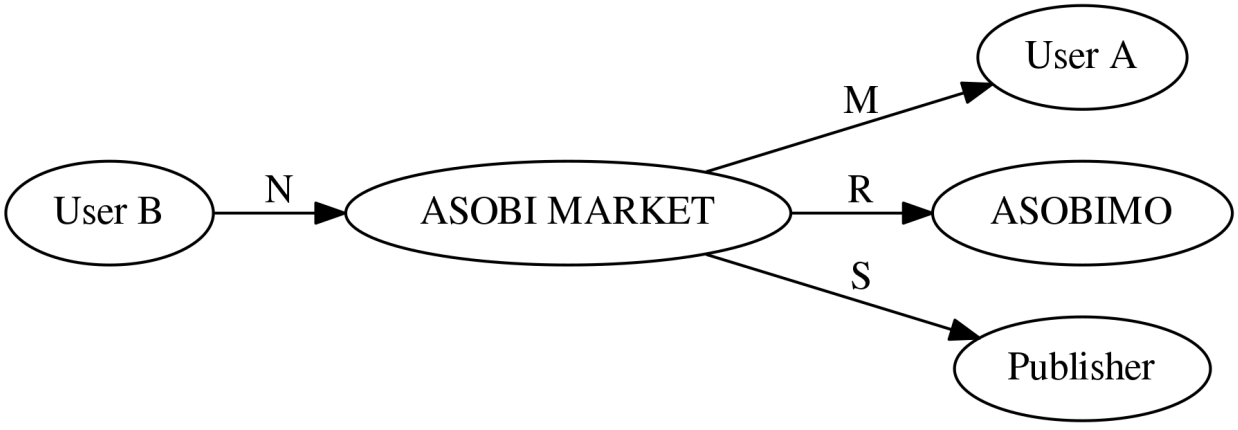

Figure 9: Flow of ASOBI COIN in Content Exchange (mediated)

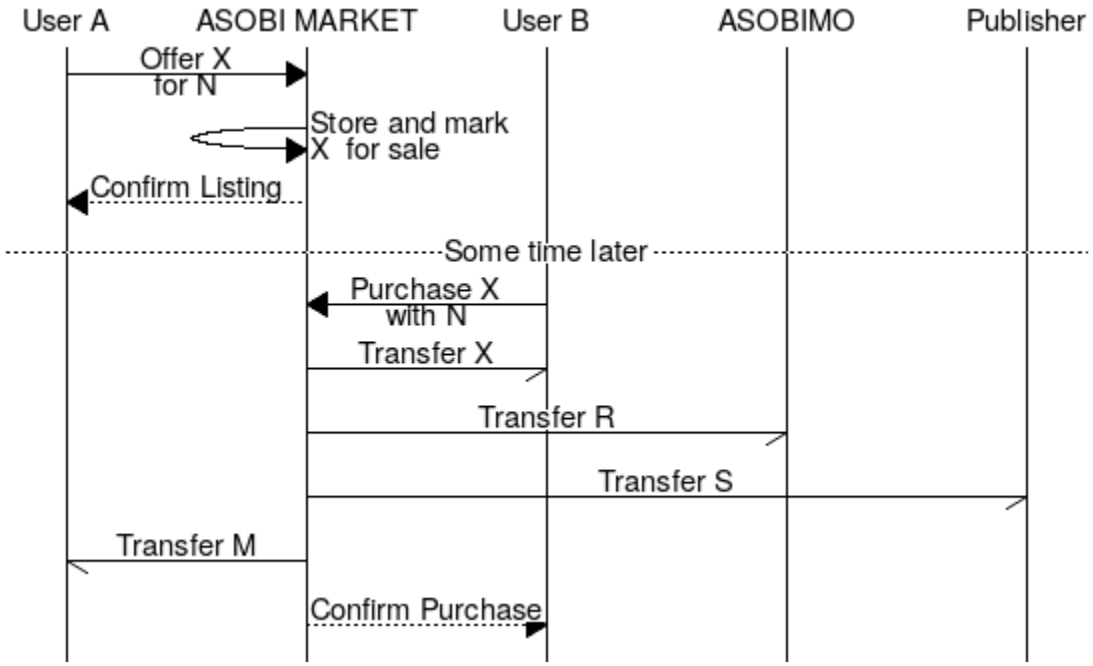

Figure 10: Content Exchange (Mediated)

7.3 Content Exchange (Peer-To-Peer)

The exchange mechanism is similar to fig. 4. Since the exchange happens without using any currency, no revenue share will result from this transaction. We assume that *User A* wants to give *User B* their license X for content Y. It depends on the business model whether this use case should be allowed. We refer to fig. 11 to understand the exact sequence of events.

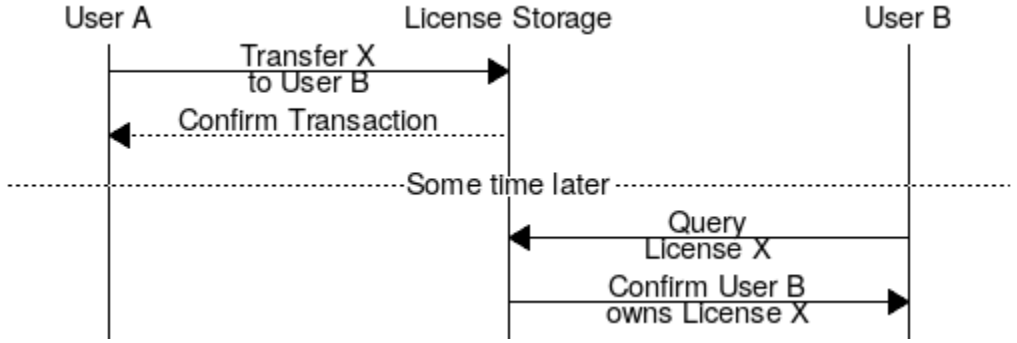

Figure 11: Content Exchange (Peer-To-Peer)

8 Viewer Application

8.1 Signing View Requests

Given a viewer client containing a user wallet and a corresponding private/public key pair, the user can create signed viewing requests. Signing the requests guarantees to any third party that the request comes indeed from the user. Signing in the case of Ethereum uses elliptic curve cryptography to create signatures

with the user's private key and confirm them using a user's public key. Without owning the private key, no one else is able to create the same signature.

Let r_X be the request to view content Y with the corresponding user license Y. Given the user's private key P_{private}, the user can use the blockchain's signing function $S(M, P_{\text{private}})$ to sign a message m using a private key P_{private} and retrieve signature M. To retrieve a signed request R_X, the client computes $S(r_X, P_{\text{private}})$ and receives R_X. Since the private key is only owned by the viewer client, no one else will be able to create R_X. Now assume that a corresponding verification method $V(m, M, P_{\text{public}})$ exists, that can take any signature M and user's public key P_{public} and confirm that indeed the user has created the signature for the corresponding message M. Then we are able to create viewing requests that can only come from the user. The DRM Server is then able to verify the request using $V(r_x, R_X, P_{\text{public}})$. See fig. 12 for the sequence of necessary steps.

It still lies in the hands of the DRM Server to verify whether the user actually owns a license to access Y, but it is impossible for any other user or third-party to create a request on behalf of the user to access Y with license X.

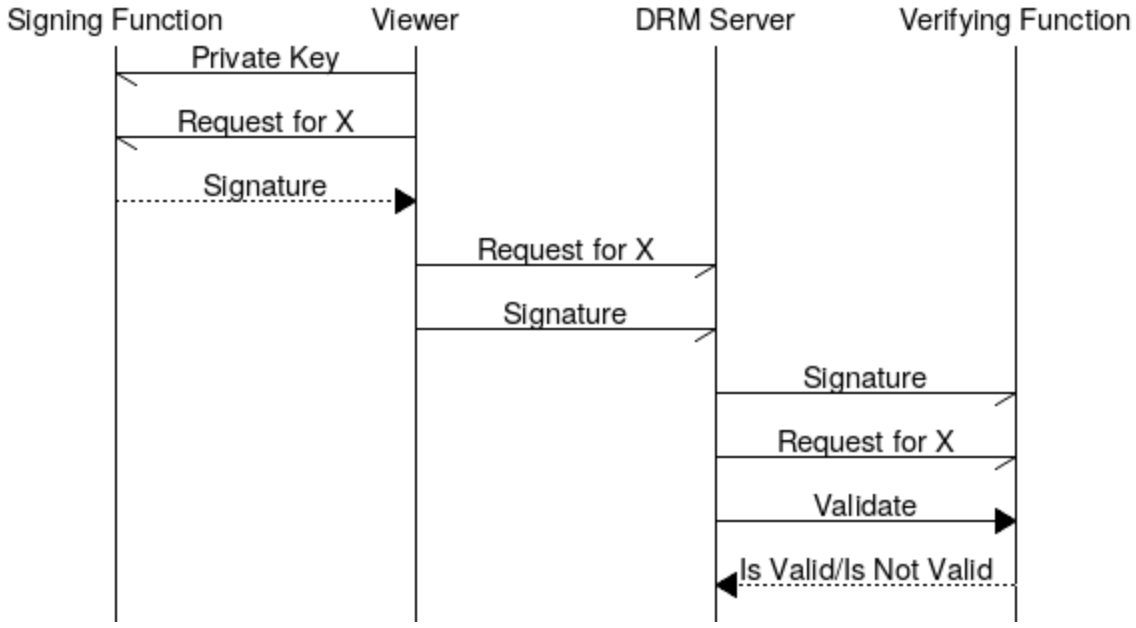

Figure 12: Signed Request Verification

8.2 Requesting Decryption Keys

Let Y be the content that a user wants to view using license X that they have previously acquired. They will use their viewer application to watch the content. Fig. 13 details sequence of steps necessary to acquire the decryption keys.

8.3 Requesting Decryption Keys (Without License)

Let Y be the content that a user wants to view. We assume that the user does not possess the necessary license to view it. They will use their viewer application to try to watch the content. Fig. 14 details sequence of steps necessary to view the content.

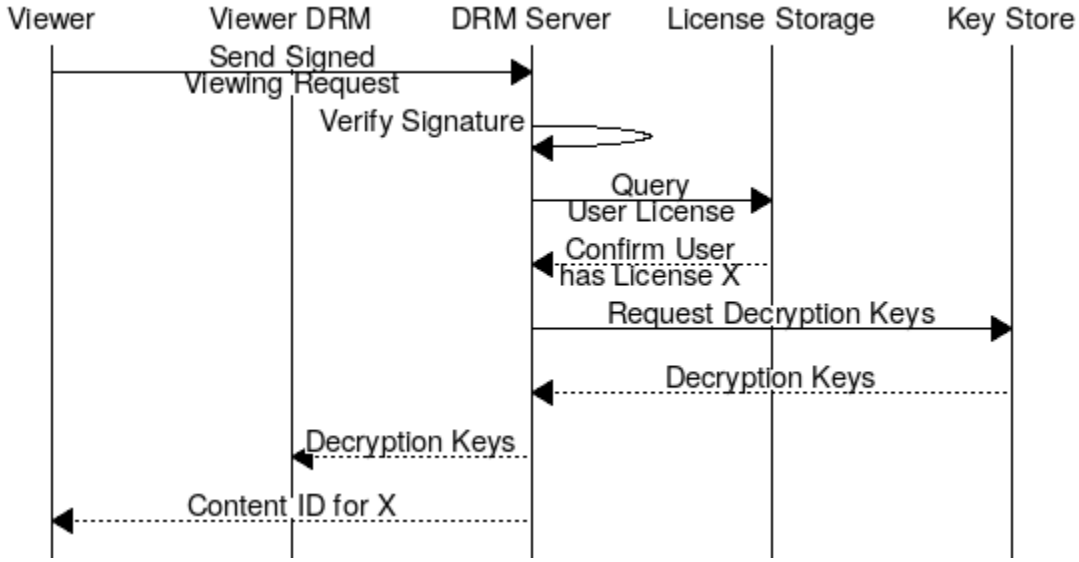

Figure 13: Requesting Decryption Keys

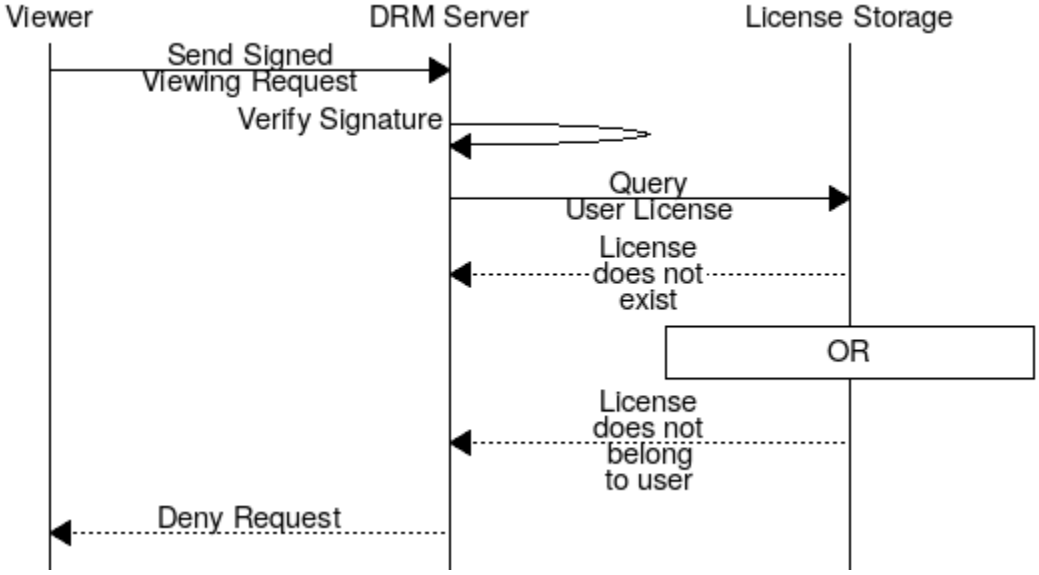

Figure 14: Requesting Decryption Keys (Without License)

8.4 Viewing Content

Assuming that the viewer DRM has successfully retrieved the decryption keys, the specific viewing process can be seen in fig. 15.

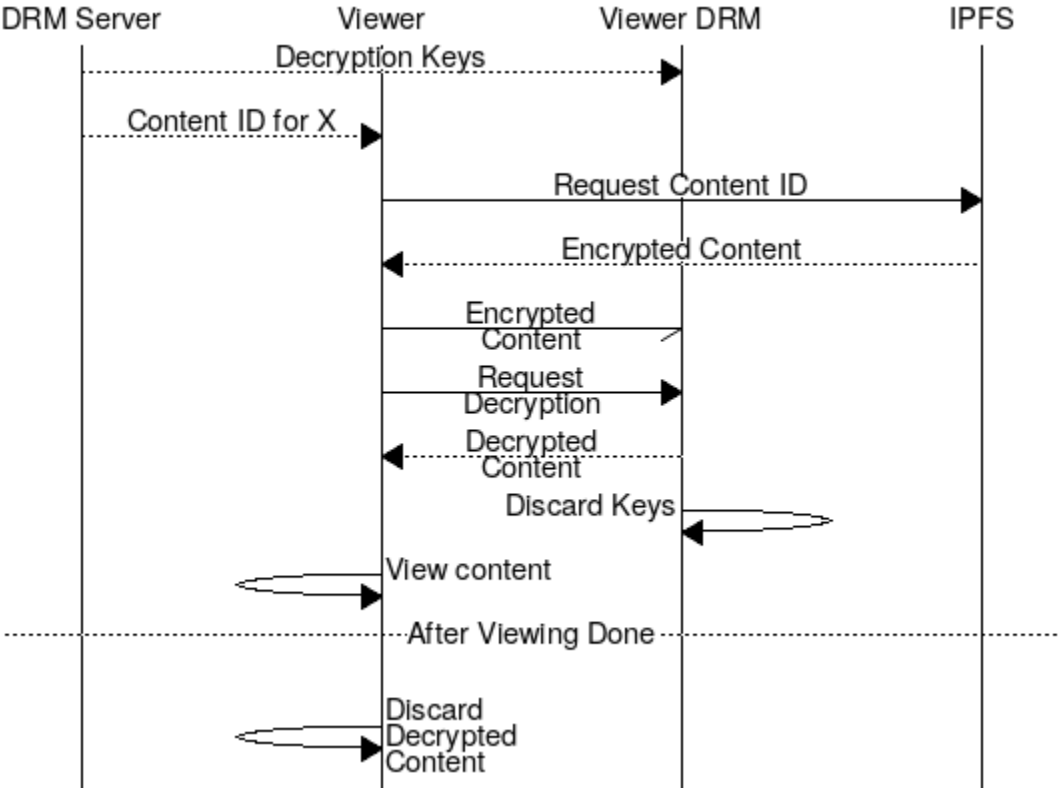

Figure 15: Viewing

9 Viewer DRM

For DRM we propose to use the open and standard Encrypted Media Extensions that have been created by Google, Microsoft and Netflix. It is widely supported by browsers and is well documented.

10 Governance and Smart Contract Ownership

In order to keep revenue shares and content creation fair between publishers and ASOBIMO, I propose using a multi-signature based governance system. For the Ethereum blockchain, there are many implementations available, such as the ConsenSys MultiSigWallet.

Multi Signature Wallets allow several Ethereum addresses to manage a smart contract together and only execute actions on it once a consensus has been reached. This is implemented using proposals and giving signers the ability to sign the proposal or vote against it. Once the proposal is accepted, a certain action on the smart contract that the multi signature wallet owns will be performed. Furthermore, signers can be added or removed using proposals. The amount of signatures that are required can also be configured. For example it can be set that a proposal will be accepted once exactly 50% of all signers have signed the request.

The signing process works similar to fig. 12.

10.1 Ownership

The ownership architecture would look as described in fig. 16. An address' ownership of a smart contract refers to the ability to

1. transfer said ownership to another address, and
2. be the only address that can execute special smart contract functions.

In the case of the License Storage, the ASOBI MARKET being the owner of it implies no one else being able to issue new licenses.

In the case of the ASOBI MARKET, the multi-signature wallet is the only address that can add content IDs or adjust revenue share settings.

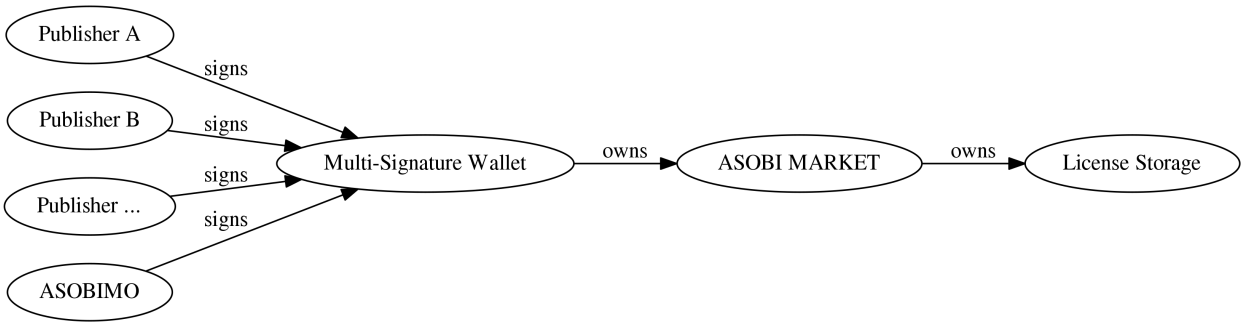

Figure 16: Ownership Architecture

10.2 Voting on Proposals

We assume that some signers want to change the revenue share settings for a certain content ID. We call this proposal *P*. In the example our multi-signature wallet has the requirement that 2 out of 3 signers need to vote for the proposal in order for it to be accepted. Moreover, we call the 3 signers in this example *Signer A*, *Signer B*, and *Signer C*.

Fig. 17 explains how the voting process works.

Furthermore, proposals can of course be denied or run out of time as in fig. 18 or fig. 19.

11 Appendix A. References

Table 7: References

Blockchain-based digital rights management (Pat US20180115416A1)	https://patents.google.com/patent/US20180115416A1/en?oq=20180115416
ERC-223	https://github.com/ethereum/eips/issues/223
ERC-777	https://github.com/ethereum/eips/issues/777
Encrypted Media Extensions	https://dvcs.w3.org/hg/html-media/raw-file/eme-v0.1/encrypted-media/encrypted-media.html

Table 7: References

MultiSigWallet	https://github.com/ConsenSys/MultiSigWallet

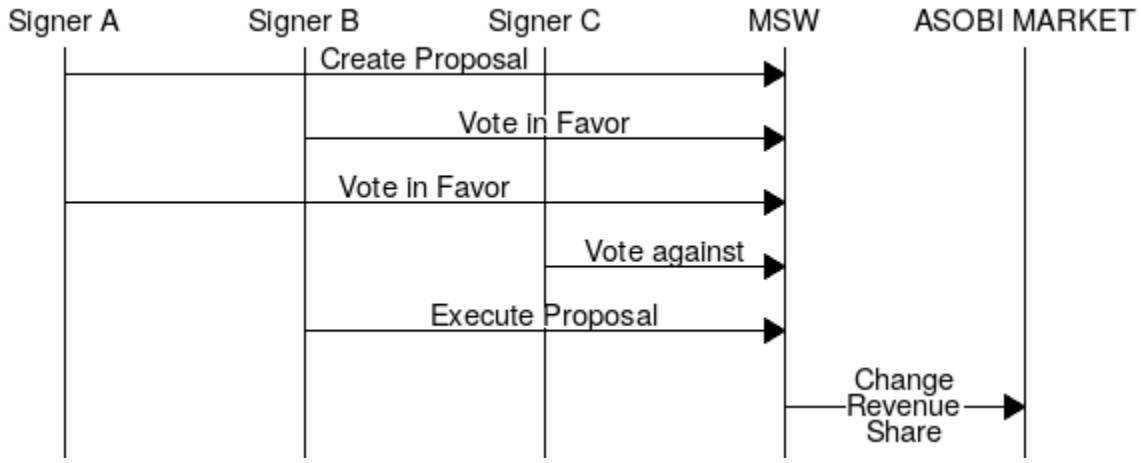

Figure 17: Voting on Proposals

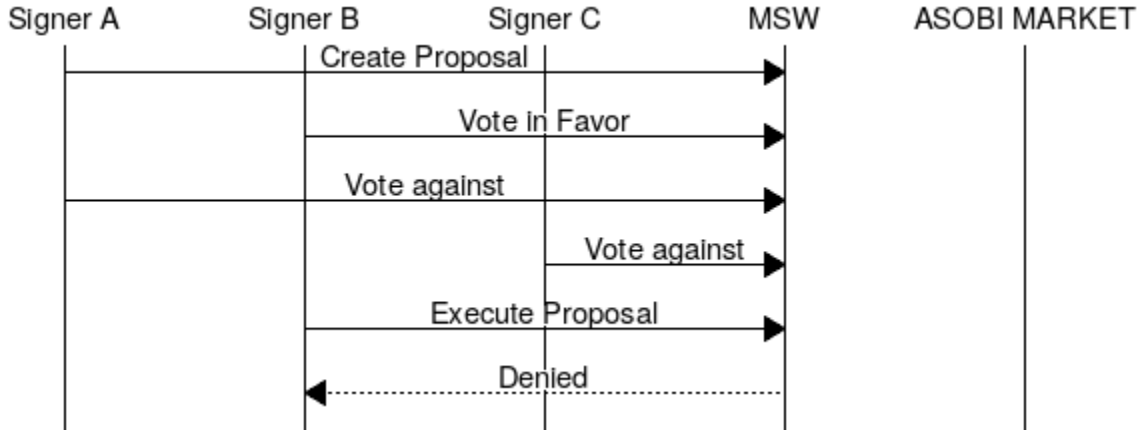

Figure 18: Voting on Proposals (Denied)

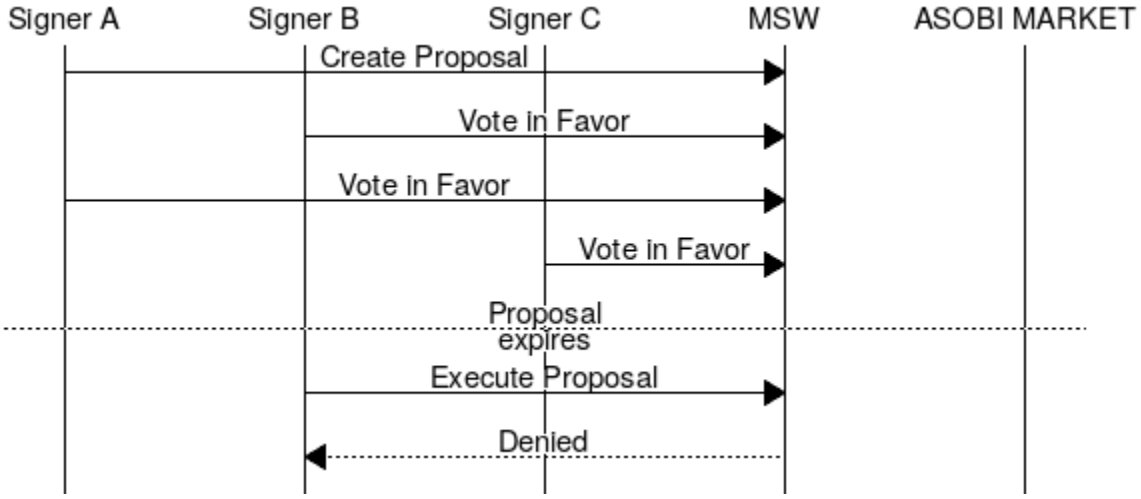

Figure 19: Voting on Proposals (Expired)

12 Appendix B. List of Figures and Tables

List of Figures

1 Architecture Overview . . . 4
2 Purchasing ASOBI COIN on DEX (Single Transaction) . . . 6
3 Purchasing ASOBI COIN on CE . . . 7
4 Transferring ASOBI COIN among peers . . . 7
5 Generic ASOBI COIN interaction (single transaction) . . . 8
6 Adding Content to CMS . . . 9
7 Removing Content from CMS . . . 9
8 Content Purchase (Single Transaction) . . . 10
9 Flow of ASOBI COIN in Content Exchange (mediated) . . . 10
10 Content Exchange (Mediated) . . . 11
11 Content Exchange (Peer-To-Peer) . . . 11
12 Signed Request Verification . . . 12
13 Requesting Decryption Keys . . . 13
14 Requesting Decryption Keys (Without License) . . . 13
15 Viewing . . . 14
16 Ownership Architecture . . . 15
17 Voting on Proposals . . . 17
18 Voting on Proposals (Denied) . . . 17
19 Voting on Proposals (Expired) . . . 17

List of Tables

1 Glossary . . . 3
2 Architecture Term Explanation . . . 3
3 ASOBI COIN: User Balances . . . 5
4 License Storage: User Licenses . . . 5
5 License Storage: Content Belonging to License . . . 5
6 Content Properties . . . 5
7 References . . . 15
7 References . . . 16