Technical Specifications for Platform Development

Contents

1. General Information about the Product

Mira – a decentralized service presented as a desktop/mobile/web application, allowing user operations with MiraBox container(s).

MiraLab.io – an online service for working with MiraBox container(s).

MiraWallet – a desktop application (macOS, Windows, Linux) for working with MiraBox container(s).

MiraWallet Mobile – a mobile application for devices based on iOS and Android.

MiraNet – the Ethereum-based Mira blockchain.

Smart contract – the essence of the Mira Blockchain containing metadata on the box and its current status (open/closed.)

MRC – the MiraNet network currency that is technically presented as ether and is used for gas payments within MiraNet; can be exchanged for Mira tokens at a 1:1 rate.

MIRA token – the currency for the MainNet; it is placed on exchanges and is given to investors as payment during the ICO. Can be exchanged for MRC at the rate of 1:1.

SmartBox Laboratory - a separate branch of the Mira platform where third-party developers can create customized smart contracts and offer them to the system.

MiraBox – a container that consists of a file and an attached contract in MiraNet.

MultiBox – a type of MiraBox whose body consists of:

- Currency #1 (one of the currencies in the container)
- Currency #2 (one of the currencies in the container)
- Currency #N (one of the currencies in the container)
- File (0<25 MB) - any type of file up to 25MB

NominalBox – a type of MiraBox whose body consists of:

- Currency - single currency in a container

SmartBox – a MultiBox with an underlying smart contract. The smart contract can restrict the opening of SmartBox contents before meeting or reaching certain external conditions, when receiving information from oracles.

Private MiraBox – has a minimum of open information: id number, opening status and public keys to the nodes used for encryption.

MiraOracles – offchain systems analyzing data that can be used as SmartBox opening conditions.

MiraNode – one of the server-end components which provides generation and storage of keys to multisignature contracts/wallets for all supported cryptocurrencies.

2. Software Requirements

2.1. Functional Requirements

Mira develops an open multicomponent service whose members can be presented as server entities as well as as client software users.

Server software should include the following components: MiraNet, MiraNode, and MiraOracle, which can be installed partially as well as fully. Owners of servers where the aforementioned services will be located, will be able to receive income from performing operations connected with supporting the Mira infrastructure.

- MiraNet (Ethereum Private Network Nodes) hosting owners - payment for transaction mining.
- MiraNode server owners - payment for MiraBox decryption.
- MiraOracle administrators - payments for successful oracle predictions.

Users can access services with the help of a browser (MiraLab), mobile phone (MiraWallet Mobile) or Windows/macOS/Linux desktop computer (MiraWallet).

- MiraBox consists of a file and contract components. The following data is stored in the container:
- MiraBox ID - identifier presented as a unique ECDSA public key
- Contract – MiraNet address of a contract which contains information about the container and its condition
- Public Data – open access information on the given container
- Private Data – contains an ECDSA private key encrypted with a symmetrical AES-256 key. The private key corresponds with the MiraBox ID.

When creating a MiraBox the corresponding contract is deployed. Its address is recorded in the open “Contract” field inside the file. Moreover, the contract can always be used to check the opening status of the container, as well as to find out what kind of data is in it.

A contract stores:

- Type – (of content)
    - MultiBox - if MiraBox also contains encrypted content that is not cryptocurrency
    - NominalBox
- Conditions - opening conditions
    - Box (MiraBox without opening conditions)
    - SmartBox (MiraBox with particular opening conditions)

- Creation Date
- Opening date
- State (status)
- Nodes - a set of addresses for nodes participating in encryption
- Currencies - a set of multisignature address(-es) of one or several currencies in the container
- Hash - sha256-hash from the encrypted part of a MiraBox file necessary to protect it (the file) from spoofing

MiraBoxes that contain only cryptocurrency and have nominal value corresponding with the content value are called NominalBox.

When creating a MiraBox, the platform generates a public key to the selected wallet to which funds can be transferred from a personal wallet or which can be used to buy cryptocurrency through the platform using fiat money.

2.2. Server Requirements

Recommended requirements for the server equipment:

- 2 GB RAM
- 2 CPU Cores
- 50GB SSD/HDD Storage

A Node should include an Ethereum node server (for example Geth) with properly set up configuration for working in private network mode.

The private network (MiraNet) should have all the Ethereum functions including the support of creation/execution of smart contracts and interaction with them.

The second important component of the server-end is the MiraNode service. It presents the main part of the server's functionality. The service provides generation and storage of keys to multisignature contracts/wallets for all supported currencies.

One more server entity, functioning as a smart-oracle, is MiraOracle. This is presented by the service monitoring MiraNet event logs and reacting to them by checking offchain-events and reporting them back to the contract through a transaction.

Interaction of server components with each other and their connection with the client software is mainly carried through a smart contract on MiraNet.

3. Description of System Components

3.1. Payment Processing

A centralized web-system that accepts fiat payments and can make cryptocurrency deposits to MiraBoxes. On one hand this is presented as the payment gateway in a similar way to (for example) www.interkassa.com, on the other hand as the minter-blockchain that deposits MRC to users after successful payment processing.

3.2. MiraWallet (macOS, Windows, Linux):

Provides the functionality for creating and opening MiraBoxes. Upon launch the application updates the Node registry and requests statuses of the containers created earlier.

Functions

1. Creating MiraBoxes:

Provides users with a choice of appropriate settings through a container configurator allowing the creation of common MiraBoxes, as well as SmartBoxes.

After entering the characteristics, a user either pays using payment processing or deposits cryptocurrency on their own. The box is then registered on the blockchain and user sets a password encrypting the content, and the application provides a MiraBox in the form of a file.

2. Opening:

In order to see MiraBox contents in a decrypted form, a user can simply transfer the file to the application window and enter the password. Then, the desktop application will in turn make a request to the Mira blockchain.

3.3. Mobile Application (iOS, Android)

Duplicates the functions of the desktop application. Has a feature for exporting MiraBox files through e-mail and messengers.

3.4. Web-application

The application's functions, including MiraBox generation should be presented mostly by scripts running on the client component except for requests to Nodes and the blockchain. Requests to nodes are executed through JSON-RPC. Integration with the MiraNet blockchain can be realized through Metamask/Mist configured on Mira or through an external web3-provider. The web application interface should be visually similar to the desktop one, however, it's layout should be adaptive to support devices with different display resolutions.

According to its functions, as well as MiraWallet, it is divided into the following sections:

1. Designing MiraBoxes, choosing the set of cryptocurrencies for building nominal containers.

   - SmartBox configurator, instruction generation for oracles.

2. Viewing MiraBoxes.

   - Viewing the status/metadata of an opened container from a file and on the blockchain.
   - Decryption and fund withdrawal from MiraBoxes.

3.5. Oracles + Nodes

Present a number of microservices: Ethereum Private Network Node, MiraOracle, and MiraNode.

3.5.1. Ethereum Private Network Node

Provides everything necessary for operational capabilities of the Ethereum-based private network including its smart contract mechanism.

3.5.2. MiraOracle

The system enabling performing of decentralized offchain operations similarly to ChainLink and Oraclize. Oracles in the system are a necessary source of external information for executing SmartBoxes.

3.5.3. MiraNode

One of the main system components. At the same level as Mira user applications. The Mira blockchain provides distributed encryption/decryption of containers of different types.

4. Classes, Roles and Characteristics of Users

Role	Features			
	MiraBox viewing and validation	NominalBox	MultiBox	SmartBox
User	+			
Authorized User	+	+	+	+
Tokenholder. Administrator	+	+	+	+
Tokenholder. Arbitrator	+	+	+	+

User

Having installed the application or after visiting miralab.io site any visitor can use the following functions without registering or logging in:

- Viewing the public information on MiraBoxes in MiraNet including nominal;
- Checking the MiraBox files' integrity and authenticity.

Authorized user

Registration and logging into the system is carried out using ECDSA key pairs which can be imported/exported between different devices.

The registration process is presented as a local generation of a pair of keys which cannot be transferred to a third party or to Mira microservices for security reasons. However, for a user's convenience they can transfer their keys to another device using QR-codes or just by copying them. An authorized user has access to all platform services and can:

- Create a MiraBox (Nominal or Multi)
- Use the SmartBox constructor.

In the mobile application an account can be further protected with a short digital pin-code. In iOS the application can be protected by Touch ID or Face ID (iPhone X).

Tokenholder-Administrator

Has the same capabilities as an authorized user, as well as the right to organize their own Mira system node with the possibility of monetization using:

- Offchain-event prediction (Oracle)
- transaction mining
- storing MiraBox keys

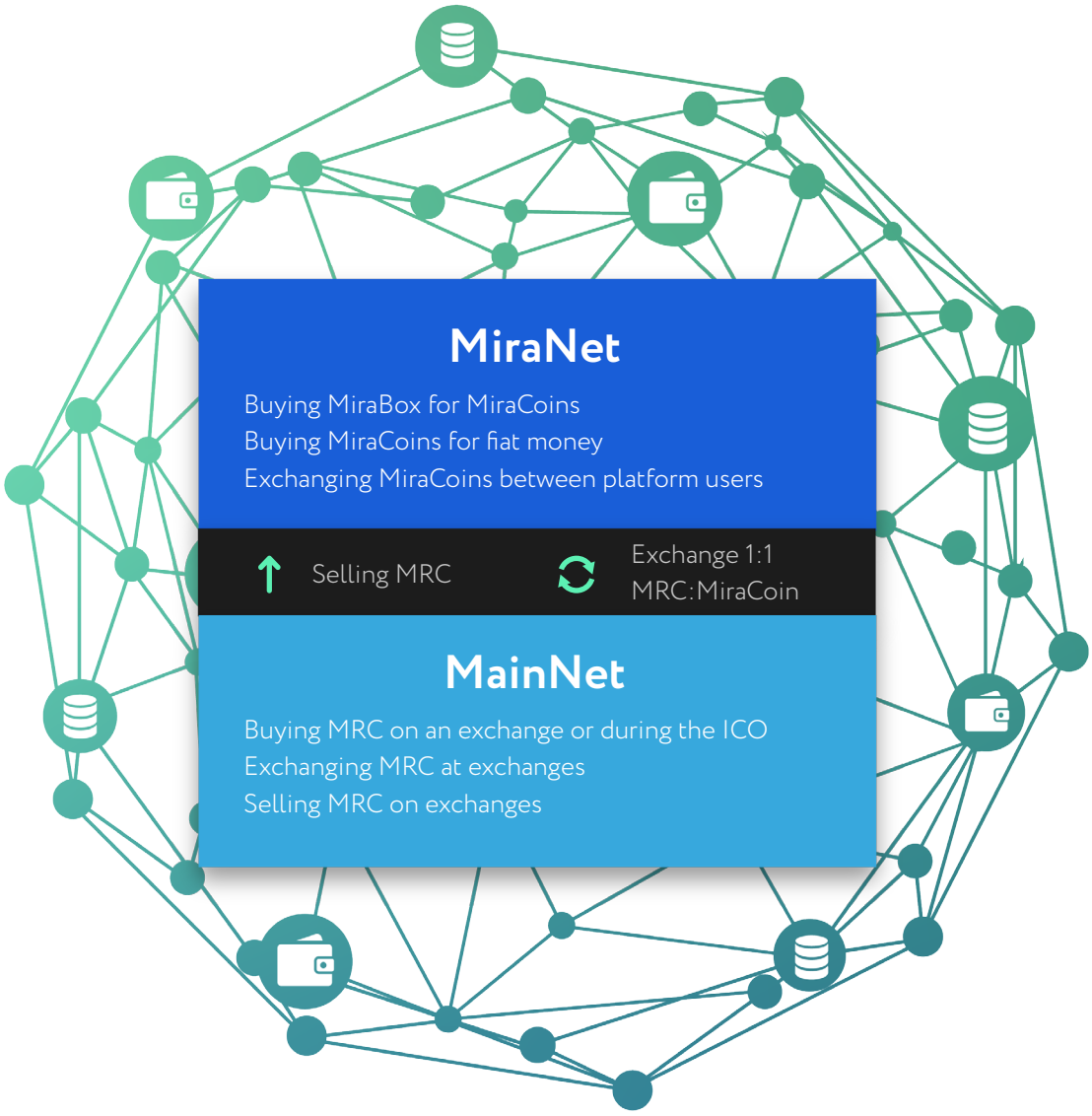
Tokenholder-Arbitrator - a user certified by the Mira platform. Has all the rights of an authorized user, as well as access to the transaction where their participation was activated. An arbitrator has the right to resolve disputed issues.

5. Mira Platform Internal Token

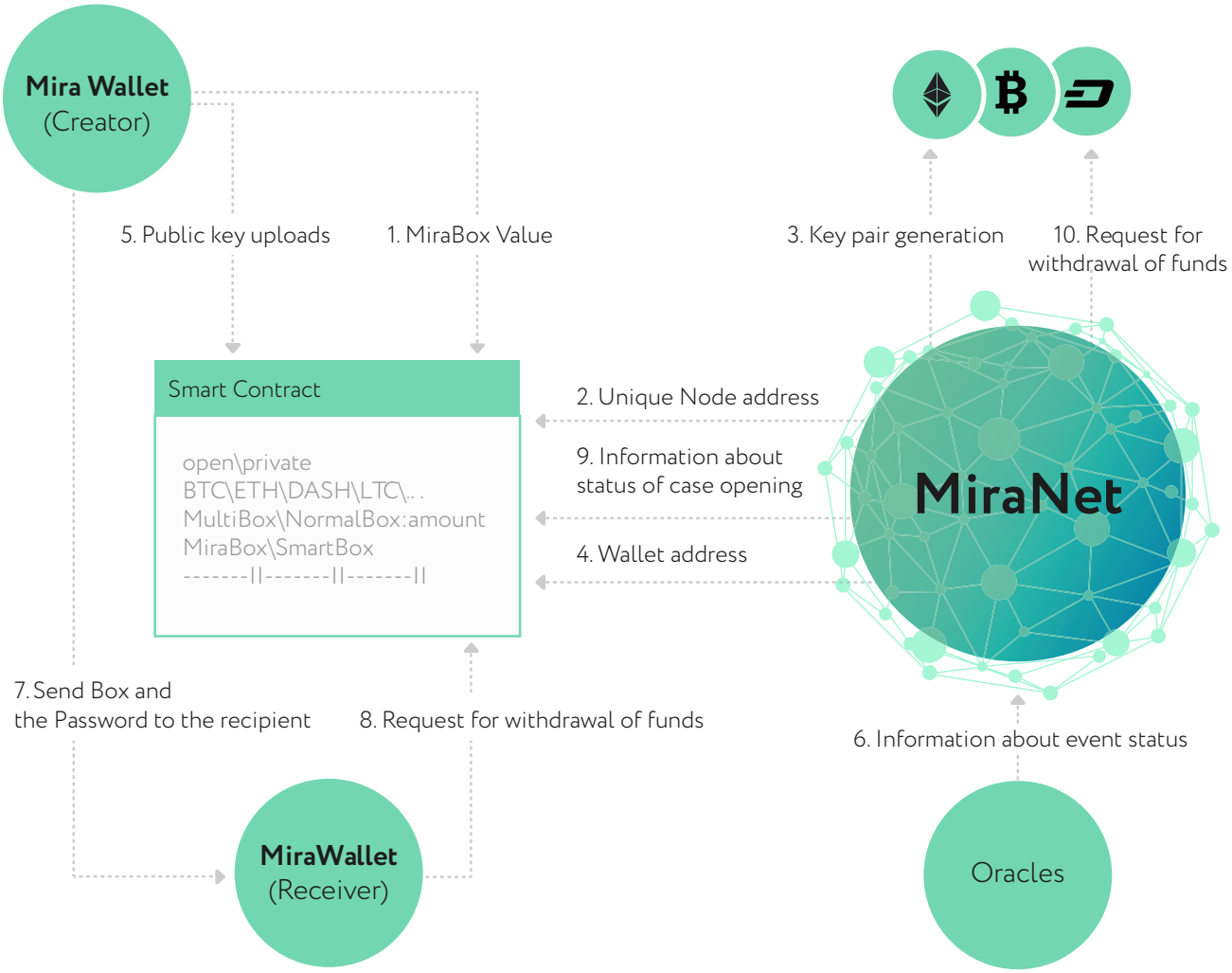
MRC - is the internal currency of the service structurally presenting as an Ether analog for the MiraNet network. MRC performs payment functions, enables buying of SmartBoxes and simultaneously serves as gas payment for executing smart contracts. MRC can be exchanged at any time for MIRA token at the rate of 1:1.

MRC can be transferred from one user to another. There are two ways to receive MRC:

1. Buying MIRA tokens during the ICO or at an exchange, then exchanging them for MRC;
2. Buying MRC for fiat money through the payment gateway.



6. User-System Interaction Scheme



1. User initiates (Smart) MiraBox creation in an application (Desktop/Web/Mobile) and configures conditions for when the Box can be opened. In addition, the smart contract responsible for a particular box is deployed, and at this time a user is charged in MRC.

At this point, the contract only describes metadata:

- openness (open/private),
- content (BTC/ETH/DASH/LTC...),
- nominal (MultiBox/NominalBox:amount),
- opening conditions (box/SmartBox)
- other data - time of creation, identifier, etc.

2. Nodes receive notification about a newly created box, the addresses of several randomly chosen nodes are saved in the smart contract, which enables finding out which nodes can be used for participating in opening the box, for which they will be rewarded with a bonus.

3. The aforementioned nodes generate and save pairs of multisignature keys of the corresponding currency/currencies.

4. The multisignature wallet address is returned to the contract.

5. The Mira application generates a pair of keys for the corresponding currency, encrypts the private key with a password, uploads the public key, and the hash from the private key is passed to the smart contract.

If (apart from the cryptocurrency) a MiraBox contains arbitrary information, i.e. the container is a MultiBox, it is also placed in the contract in encrypted form. Data is asymmetrically encrypted using ECC with the public keys already saved in the smart contract in step 3 and symmetrically using AES-26, user password.

6. Oracles record the occurrence of the determined event in the contract if it is a SmartBox.

7. The creator of the box transfers it to the recipient together with the password.

8. The recipient opens the Box file in the Mira application, enters the password, and if the password is correct the system is able to extract a private key which is used to call the contract which enables withdrawal of funds from MiraBox.

9. Nodes receive the data regarding successful opening from the blockchain.

10. Nodes using the private keys created during step 3 make a request to the multisignature wallet to withdraw funds. If all the signatures are correct the funds are successfully withdrawn. If it is a MultiBox with arbitrary information, the nodes also pass the generated private keys to the contract for decryption of this information.

7. Security

The Mira platform is a decentralized service, thanks to which any potential unauthorized access to any infrastructural elements will not allow access to MiraBox content. The following cryptographic algorithms are used for encryption: AES-256 and ECC-Secp256k1. These algorithms provide robust protection for MiraBox files from local bruteforce.

All the system components are opensource-solutions whose source code will be available on GitHub and can be audited by anyone.

Interaction between microservices whose API is presented as REST (for example, JSON-RPC) is carried out using only HTTPS with a trusted, signed SSL certificate.

7.1. Multisignature Mechanism

To open a MultiBox containing cryptocurrency, it is necessary to undergo multisig-verification. This means that more than one ECDSA signature is needed to make a transaction.

So, no member of the system can gain access to the tools unilaterally. At the same time after opening, a MiraBox is marked as used and cannot be used again in a MiraWallet.

7.2. MiraBox File Double Encryption

When creating a MiraBox through MiraWallet, its content is encrypted with the help of the AES-26 symmetrical algorithm. If a user creates a MultiBцox, then additionally content is encrypted with asymmetrical ECC (Elliptic-curve cryptography.)

7.3. Oracles

A distributed system of predicting offchain-events, built on the same principle as the Oraclize and ChainLink services. That is, events are checked by multiple independent servers with a reputation-rating system which is used to carry out PoS (Proof-of-stake) confirmation of an oracle's prediction.