

Bilkent University
Department of Computer Engineering

CoinAMI

Coin-Application Mediator Interface

Supervisor

Can Alkan

Members

Ahmet Kerim Şenol

Alper Gündoğdu

Halil İbrahim Özercan

Muhammed Yusuf Özkaya

Jury Members

Buğra Gedik

Cevdet Aykanat

Analysis Report
November 3, 2014

Contents

1	Introduction	1
2	Current System	2
2.1	DNA Sequence Alignment System	2
2.2	Bitcoin	2
3	Proposed System	3
3.1	Overview	3
3.2	Functional Requirements	4
3.3	Nonfunctional Requirements	5
3.4	System Models	6
3.4.1	Scenarios	6
3.4.1.1	Alper creates a Wallet ID	6
3.4.1.2	Alper restores his Wallet	6
3.4.1.3	Yusuf starts mining	6
3.4.1.4	Ahmet sends coin to Halil	6
3.4.1.5	Can adds Ahmet as a contact	6
3.4.1.6	Bugra sets a password	7
3.4.1.7	Halil views/exports transaction history	7
3.4.1.8	Admin adds new genome data	7
3.4.1.9	Admin collects the result data	7
3.4.2	Use Case Models	8
3.4.2.1	Create a Wallet ID	8
3.4.2.2	Restore Existing Wallet	9
3.4.2.3	Set Password	10
3.4.2.4	Send Coin	11
3.4.2.5	Add a Contact	12
3.4.2.6	Remove a Contact	13
3.4.2.7	View a Contact	14
3.4.2.8	View Transaction History	15
3.4.2.9	Export Transaction History	16
3.4.2.10	Input New Genome Data	17
3.4.2.11	Collect Result Data	18
3.4.2.12	Inspect the Statistics	19
3.4.3	Object and Class Model	20
3.4.3.1	Class Diagram for Server	20
3.4.3.2	Class Diagram for Client	21
3.4.3.3	Class Diagram for Miner	22
3.4.4	Dynamic Models	23

3.4.4.1	Sequence Diagrams	23
3.4.4.1.1	Creating a Wallet	23
3.4.4.1.2	Sending Coins	24
3.4.4.1.3	Uploading new Genome Data	25
3.4.4.2	Activity Diagrams	26
3.4.4.2.1	Creating a Wallet	26
3.4.4.2.2	Sending Coins	26
3.4.4.2.3	Exporting Transaction History	27
3.4.5	User Interface	28
3.4.5.1	Creating/Restoring a Wallet	28
3.4.5.2	Mining Coins	30
3.4.5.3	Sending Coins	31
3.4.5.4	Receiving Coins	32
3.4.5.5	Viewing Contacts	33
3.4.5.6	Viewing History	34
3.4.5.7	Options	35
3.4.5.8	Administration	36

4 References

1 Introduction

Cryptocurrencies lately acquired considerable attention from the public. First cryptocurrency was Bitcoin that was created in 2009 by an unknown person with pseudonym Satoshi Nakamoto[1]. Its aim was creating a distributed currency that is free from any controlling authority. Because of its success, other cryptocurrencies are proposed afterwards such as Litecoin[2] Peercoin[3] and Dogecoin[4] but power of their mining network is negligible compared to the Bitcoin's.

Bitcoin mining network's total computation power is 77.46 PetaHash/s[5]. In comparison, world's most powerful scientific computation grid BOINC (Berkeley Open Infrastructure for Network Computing) has a computation power of 7.54 PetaFLOP/s[6] because hashing uses integer operations and BOINC's power measured in floating point operations, even 1 to 10 scaling makes Bitcoin network as powerful as BOINC network. This computation power mainly used to maintain currency's integrity. We propose a system where this computation power will be used for scientific computation as well as integrity purposes. We chose genome sequence alignment problem for our initial proposal.

2 Current System

2.1 DNA Sequence Alignment System

DNA sequence alignment problem is being researched widely and currently there are many approaches for a fast algorithmic solution. Burrows-Wheeler method is a popular method. It takes 30 CPU days to align an individual's genome using this method. <http://bio-bwa.sourceforge.net/>

2.2 Bitcoin

Bitcoin is a secure digital currency that users can easily trade between and earn using their CPU power (mining operation). The transactions are being handled by users as well and this earns them money. Merely using CPU power for earning money is a great motivator for people to use this system, and the resulting CPU power can be used in solving many different problems with different natures.

3 Proposed System

3.1 Overview

The traditional approach, using only one CPU, still takes relatively long time to align an individual's DNA. Cryptocurrency mining process provides us with a lot of CPU power and we can make use of this to convert the computation process into distributed process.

To achieve this, we will divide each person's genome data into parts, called sub-problems, send them to different clients that will solve that sub-problems for us. Then we will fetch the results from the user, verify the correctness and integrity of the results, and grant the user some amount of coins if everything is valid. The amount of coins granted will be proportional to the size of the sub-problem sent to the user.

If the sub-problems sent to a client are all from the same person, a client can get DNA information of the person and this is a privacy violation. To avoid this, sub-problems consist of 2 or more people with anonymous data. Since the client doesn't know which part is whose DNA, he can not build the DNA data for a specific person. This adds extra work at server compared to the single person approach. Server knows which part of the sub-problem belong to whom and should arrange the data for each person after a result set is received from a client. But this arrangement is a trivial operation compared to the actual alignment operation performed by the clients and can be handled by the server.

Also users can try to forge fake result sets and send them to servers to earn money without using much CPU power. So the server should check the integrity and the correctness of the result set. To achieve this, when sending a sub-problem, server adds a little amounts of fake decoy data that is not from an actual person to the sub-problem. It knows where the decoy parts will align before sending the data and when the result-set is returned from the user, it will check if they were aligned correctly. And if any of the decoy data fails the test the user will not be granted coins. Since the user doesn't know which parts of the sub-problem are decoys, he needs to actually solve all the parts correctly. This approach prevents users from forging fake results, and if a person tries to do this for a number of times, the user will be banned from the system automatically.

3.2 Functional Requirements

1. The user should be able to make coin transactions.
2. The user should be able to earn coins through mining client.
3. People can create a new Wallet ID to use the system.
4. The user should be able to restore an existing Wallet.
5. The user can inspect and export the transaction history of defined Wallet.
6. The user should be able to manage a contacts list.
7. Server side administrator should be able to add new genome sequences to the system.
8. Server has to distribute the genome's content among users such that the divided content could not be rebuilt. This is needed to protect the human rights of genome owners.
9. Server should be able to distinguish any fake result from actually calculated results.
10. Server should be able to merge received data according to appropriate genome sequence.
11. Each completed genome data should be preserved correctly.
12. Server side administrator should be able to collect the preserved result.

3.3 Nonfunctional Requirements

Non functional requirements are one of the most important aspects of server side program. Validation of a mining process, transactions between users and creating new Wallets are should be done by server. This puts a heavy weight on server and this must be handled accurately so that Users(clients) do not feel any inconvenience in terms of usage. There are some non functional requirements for our project such as:

1. The validation of the results from user should take a small amount of time, not overwhelming the server side with validation process.
2. Transactions should be user friendly, efficient and simple so that user does not feel confused.

3.4 System Models

3.4.1 Scenarios

3.4.1.1 Alper creates a Wallet ID

Alper already downloaded the coin management client from the website of the cryptocurrency and runs the client application. At setup screen he sees a wallet creation form and requests a new Wallet ID. After client requesting and getting the Wallet IDs from the server, client asks if Alper wants to set a password to secure his wallet. Alper selects a password, the client starts up and lists the options that the user can do.

3.4.1.2 Alper restores his Wallet

Alper already downloaded the coin management client from the website of the cryptocurrency and runs the client application. At setup screen he sees a wallet creation form and chooses to restore his old Wallet ID from its backup files. The client starts up and lists the options that the user can do.

3.4.1.3 Yusuf starts mining

Yusuf starts the mining client and enters his Wallet ID which the coins will be collected. After pressing the “Start Mining” button, mining program starts mining in background.

3.4.1.4 Ahmet sends coin to Halil

Ahmet opens Coin management client and chooses send coin option. Application redirects him to coin sending form. Ahmet fills the form with amount and information of Halil’s Wallet.

3.4.1.5 Can adds Ahmet as a contact

Can opens contacts list from coin management client. Then he chooses “Add a Contact” option. Fills the form with Ahmet’s Wallet information. Finally, he submits the form and adds

a contact successfully.

3.4.1.6 Bugra sets a password

Bugra forgot to set a password for his Wallet at start up and now he wants to secure his Wallet. He starts Coin Management Client and chooses “Set Password” option from menu. Application shows a form to Bugra to enter his new password. Bugra submits the form and client updates his Wallet.

3.4.1.7 Halil views/exports transaction history

Halil opens his client application and selects View History then clicks Export button. Then a save file dialog is opened by the application. Halil types a file name. Finally, a file with csv format is created as exported data.

3.4.1.8 Admin adds new genome data

Admin opens the server’s user interface and uploads the genome files using the select file option and pressing the upload button

3.4.1.9 Admin collects the result data

Admin opens the server’s user interface, presses the collect results button and selects the folder that the results will be exported.

3.4.2 Use Case Models

3.4.2.1 Create a Wallet ID

Use Case Name: Create a Wallet ID

Primary Actor: User

Stakeholders and Interests: User wants to create a Wallet ID to use for coin transactions.

Entry Condition: User successfully started the application without any pre-defined Wallet ID.

Exit Condition: User gets a new Wallet ID

Event Flow:

1. User starts the application for first time.
2. User chooses to create a new Wallet ID.
3. Application stores the new Wallet as default.
4. Application shows the details of new Wallet.

Alternative Flow of Event 1:

1. User starts the application for first time.
2. User chooses to create a new Wallet ID with password.
3. User enters a password .
4. User re-enters the password to confirm.
5. Application stores the new Wallet ID as default.
6. Application shows the details of new Wallet ID.

Alternative Flow of Event 2:

1. Client application cannot reach the server due to a problem.
2. An error message is shown.

3.4.2.2 Restore Existing Wallet

Use Case Name: Restore existing Wallet

Primary Actor: User

Stakeholders and Interests: User wants to restore his/her existing Wallet to use as default Wallet in Coin Management Client.

Entry Condition: User has an existing Wallet ID file which he/she wants to restore.

Exit Condition: User successfully restores a Wallet ID.

Event Flow:

1. User starts the application without a pre-defined Wallet ID.
2. User chooses “Restore an existing Wallet ID” option.
3. Application redirects to a file chooser dialog.
4. User chooses old Wallet file.
5. Application restores the old Wallet and sets as default.
6. Application shows the information about Wallet ID.

Alternative Flow of Event 1:

1. User selects a Wallet file which is password protected.
2. Application asks the password to continue recovery process.
3. User enters the password.
4. Application restores the old Wallet and sets as default.
5. Application shows the information about Wallet ID.

Alternative Flow of Event 2:

1. User selects a corrupted Wallet file or does not know the password of a password protected Wallet.
2. User could not fulfil the requirements. Therefore an appropriate error message is given by application.

3.4.2.3 Set Password

Use Case Name: Set Password

Primary Actor: User

Stakeholders and Interests: User wants to set a new password or change a password defined earlier.

Entry Condition: User has a default Wallet defined in the application.

Exit Condition: Password of the Wallet has been changed.

Event Flow:

1. User chooses “Options” from menu.
2. User navigates to options screen.
3. User clicks to “Set Password”.
4. Application displays a password dialog.
5. User enters new password twice.
6. Application sets the new password and displays a success message.

Alternative Flow of Event 1:

1. User chooses “Set Password” option from menu
2. Wallet is already password protected. Therefore, new password dialog also asks for earlier password.
3. Application displays a password dialog.
4. User enters the old and new password.
5. Application sets the new password and displays a success message.

Alternative Flow of Event 2:

1. User does not enter the same password twice for new password or User cannot remember an earlier password if there is any.
2. Application shows an appropriate error message.

3.4.2.4 Send Coin

Use Case Name: Send Coin

Primary Actor: User

Stakeholders and Interests: User wants to transfer some amount of coin to someone else.

Entry Condition: User has a default Wallet defined in the application.

Exit Condition: User has sent coin successfully.

Event Flow:

1. User chooses “Send Coin” option from menu.
2. User fills the transaction form with receiver’s information, amount of coins to transfer, description and etc.
3. User clicks to “Send”.
4. User sees a dialog about the details of transaction.
5. User returns to transaction history screen.

Alternative Flow of Event 1:

1. User does not have the specified amount of coin or enters wrong information about receiver.
2. Application displays an appropriate error message about what was wrong.
3. User returns to send coin screen.

3.4.2.5 Add a Contact

Use Case Name: Add a Contact

Primary Actor: User

Stakeholders and Interests: User wants to add a contact to his/her contact list.

Entry Condition: User has a default Wallet defined in the application.

Exit Condition: User has added a contact successfully.

Event Flow:

1. User chooses “Manage Contacts” option from menu.
2. User navigates to contacts screen.
3. User clicks to “New Contact”.
4. User fills the new contact form with essential information.
5. User clicks to “Add”.
6. User returns to contacts list screen.

Alternative Flow of Event 1:

1. User fills the form with incorrect information.
2. User sees an error message about what was incorrect.
3. User returns to new contact form.

Alternative Flow of Event 2:

1. User clicks “Dismiss” in form.
2. User returns to contacts screen.

3.4.2.6 Remove a Contact

Use Case Name: Remove a Contact

Primary Actor: User

Stakeholders and Interests: User wants to remove a contact to his/her contact list.

Entry Condition: User has a default Wallet defined in the application.

Exit Condition: User has removed a contact successfully.

Event Flow:

1. User chooses “Manage Contacts” option from menu.
2. User navigates to contacts screen.
3. User clicks on a contact from contacts list.
4. User click to “Remove Contact” option.
5. A verification dialog appears and asks if the user is sure to remove the contact.
6. User confirms the removing process.
7. Contact is removed and user returns to contacts screen.

3.4.2.7 View a Contact

Use Case Name: View a Contact

Primary Actor: User

Stakeholders and Interests: User wants to see details of a contact.

Entry Condition: User has a default Wallet defined in the application and at least one contact.

Exit Condition: User has seen the details of a contact.

Event Flow:

1. User chooses “Manage Contact” option from menu.
2. User navigates to contacts screen.
3. User clicks on a contact from contacts list.
4. User click to “View Details” option.
5. A contact information dialog appears with the details about the chosen contact.
6. User inspects the contact.
7. User closes the dialog and returns to contacts screen.

3.4.2.8 View Transaction History

Use Case Name: View Transaction History

Primary Actor: User

Stakeholders and Interests: User wants to see details of transactions made.

Entry Condition: User has a default Wallet defined in the application.

Exit Condition: User has seen the transaction history.

Event Flow:

1. User chooses “Manage Transactions” option from menu.
2. User navigates to transaction history screen.
3. User sees the every transaction with detailed information.

3.4.2.9 Export Transaction History

Use Case Name: Export Transaction History

Primary Actor: User

Stakeholders and Interests: User wants to export transactions history to a file.

Entry Condition: User has a default Wallet defined in the application.

Exit Condition: User has exported the transaction history to a file.

Event Flow:

1. User chooses “Manage Transactions” option from menu.
2. User navigates to transaction history screen.
3. User clicks to “Export History”.
4. User selects a directory for the file which is going to be exported.
5. Export is done successfully and user sees a success message.
6. User returns to transaction history screen.

Alternative Flow of Event 1:

1. User clicks to “Export History” but there is no transaction in the history of Wallet.
2. User sees an appropriate warning message.
3. User returns to transaction history screen.

3.4.2.10 Input New Genome Data

Use Case Name: Input New Genome Data

Primary Actor: Administrator

Stakeholders and Interests: Administrator wants to add new genome data to the system.

Entry Condition: Administrator has new genome data as a file.

Exit Condition: Administrator has imported the genome data to the system.

Event Flow:

1. Administrator opens the interface in server.
2. Administrator navigates to “Add new genome data“ screen.
3. Administrator selects the file which contains genome data.
4. Importing is done successfully and he sees a success message.

3.4.2.11 Collect Result Data

Use Case Name: Collect Result Data

Primary Actor: Administrator

Stakeholders and Interests: Administrator wants to collect result data from the system.

Entry Condition: Application has new results collected from clients.

Exit Condition: Administrator has exported the result data from the system.

Event Flow:

1. Administrator opens the interface in server.
2. Administrator navigates to “Collect result data“ screen.
3. Administrator selects the folder which the results will be exported.
4. Exporting is done successfully and he sees a success message.

3.4.2.12 Inspect the Statistics

Use Case Name: Inspect the Statistics

Primary Actor: Administrator

Stakeholders and Interests: Administrator wants to inspect statistics about the sequenced genome data and the clients connected.

Entry Condition: None.

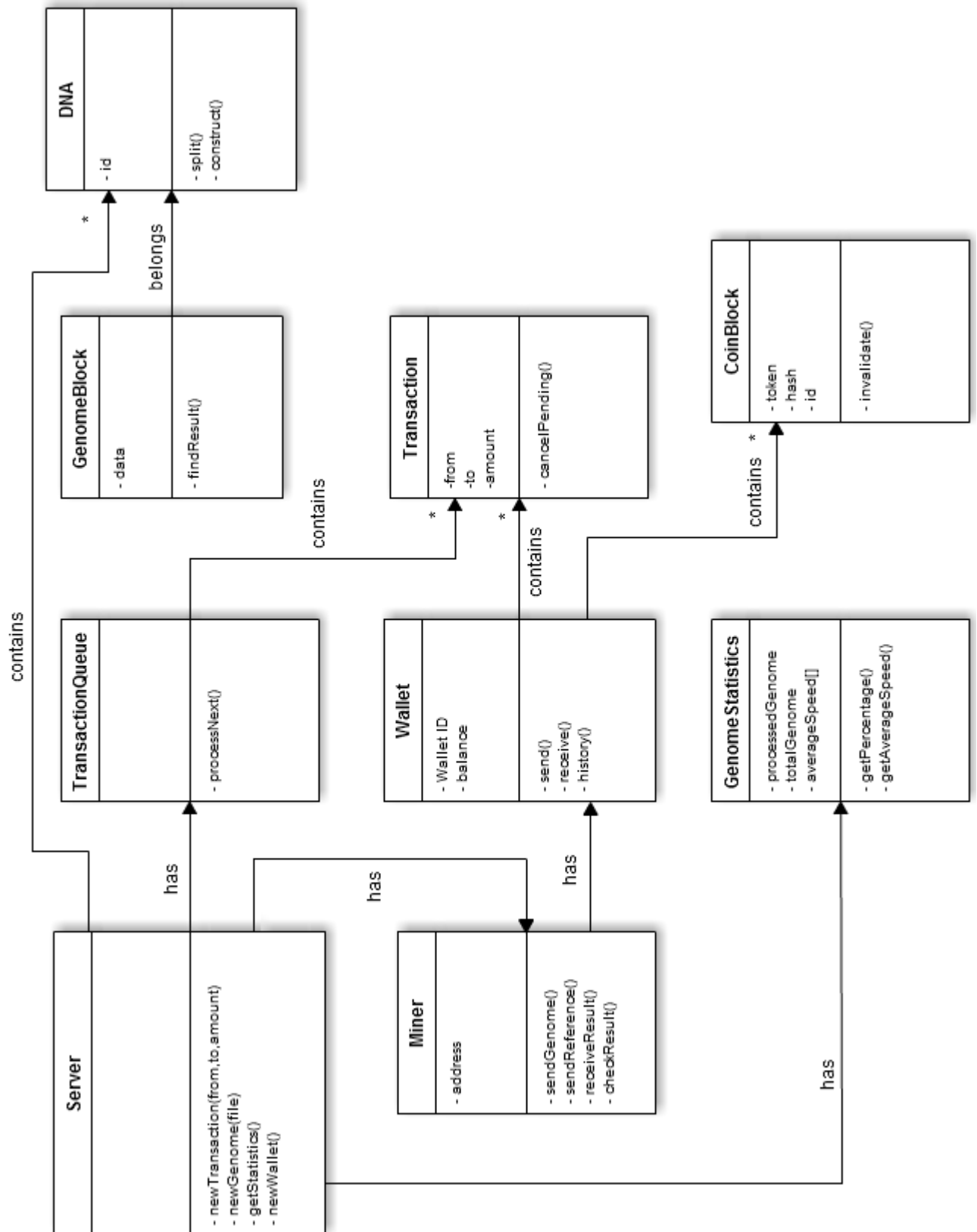
Exit Condition: Administrator has seen the statistics about the system.

Event Flow:

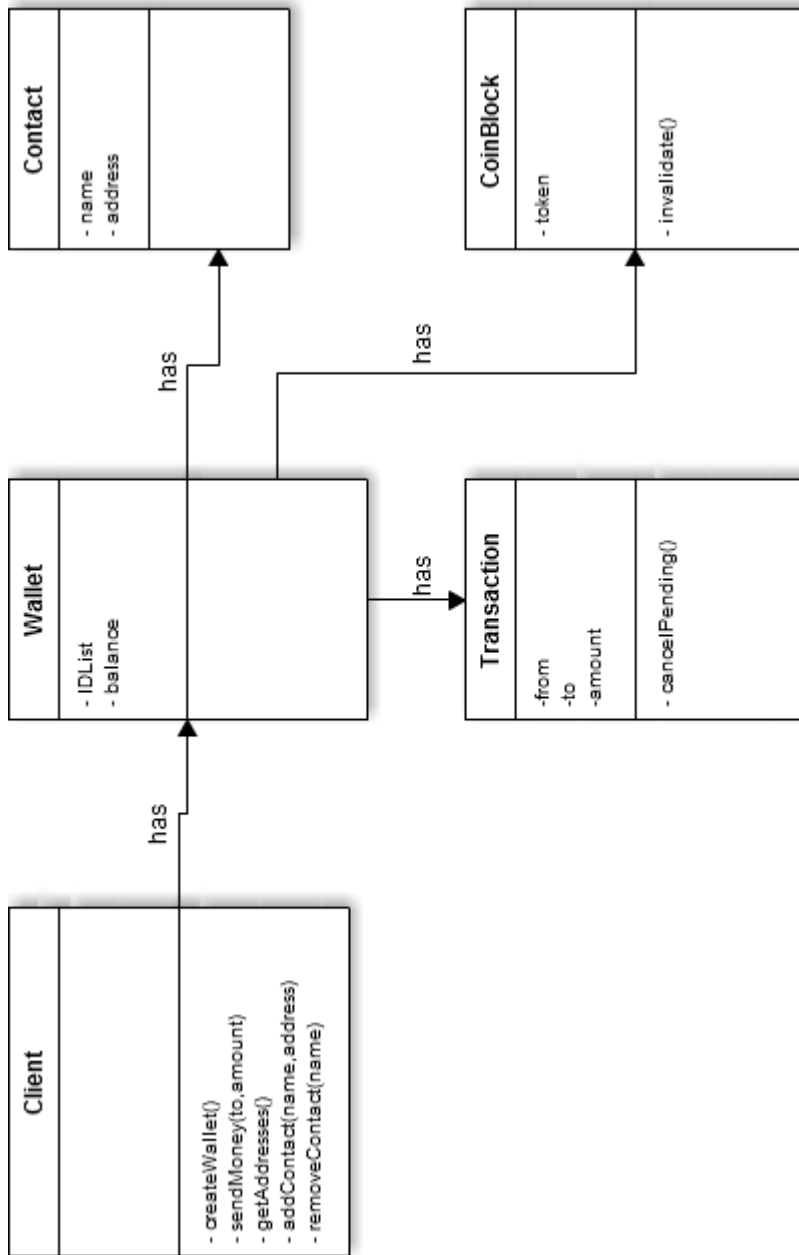
1. Administrator opens the interface installed in server.
2. Administrator navigates to “Statistics“ screen.
3. Statistics are displayed.

3.4.3 Object and Class Model

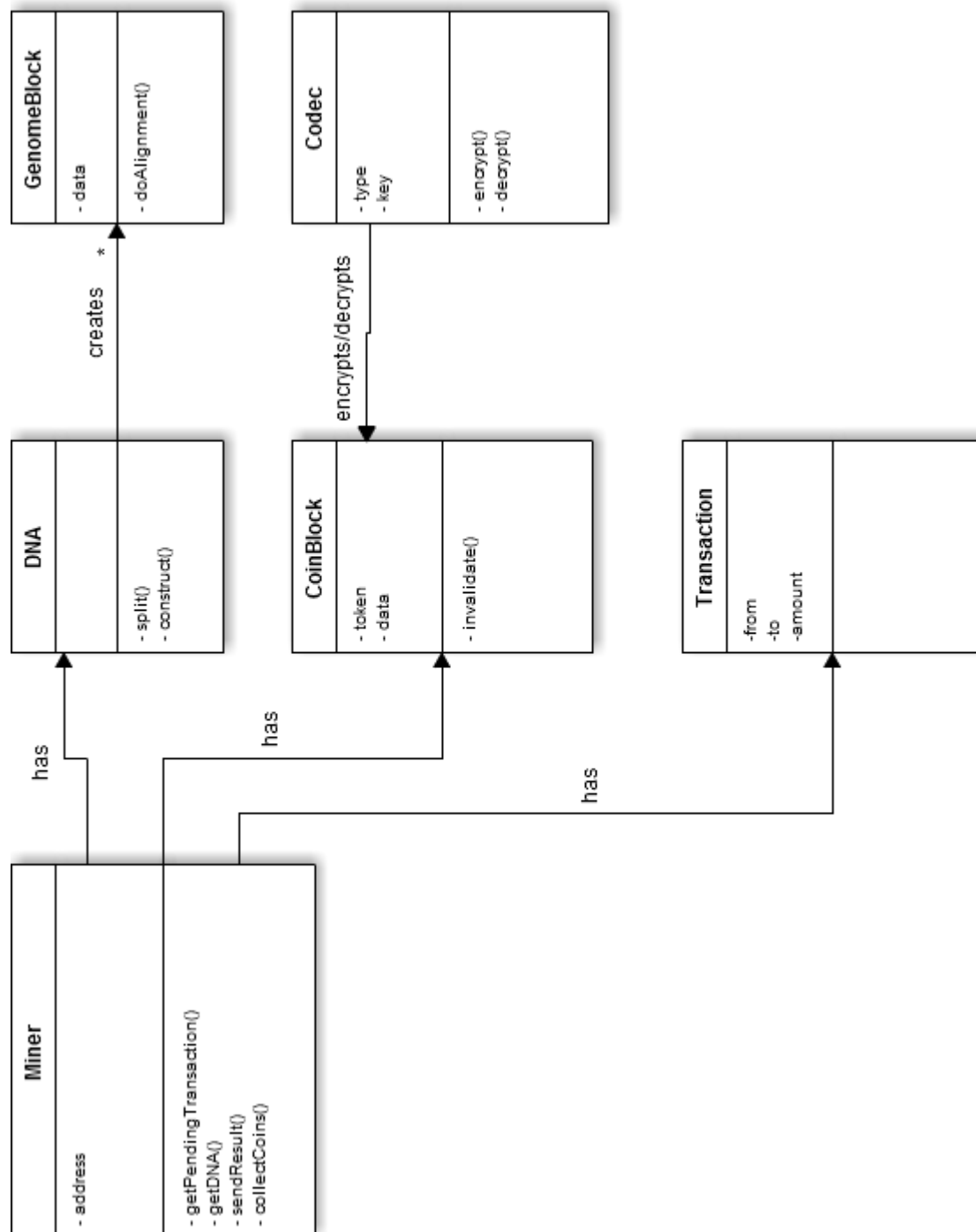
3.4.3.1 Class Diagram for Server



3.4.3.2 Class Diagram for Client



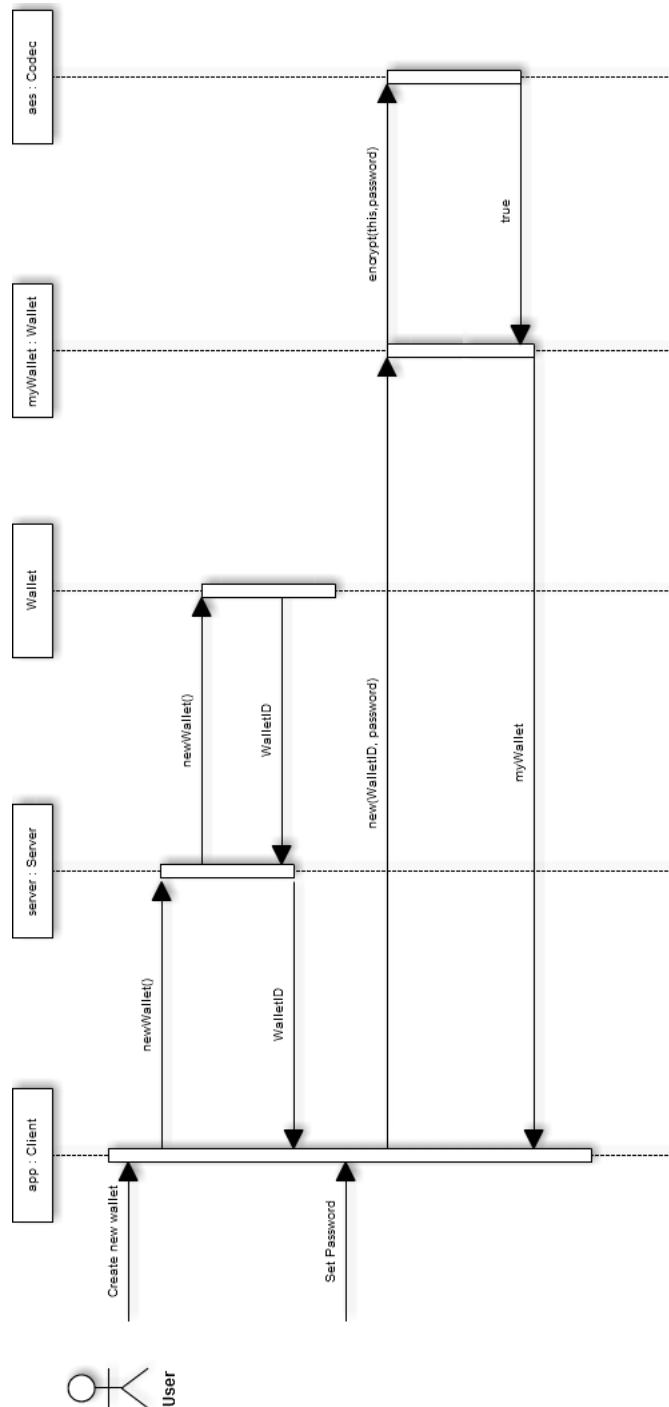
3.4.3.3 Class Diagram for Miner



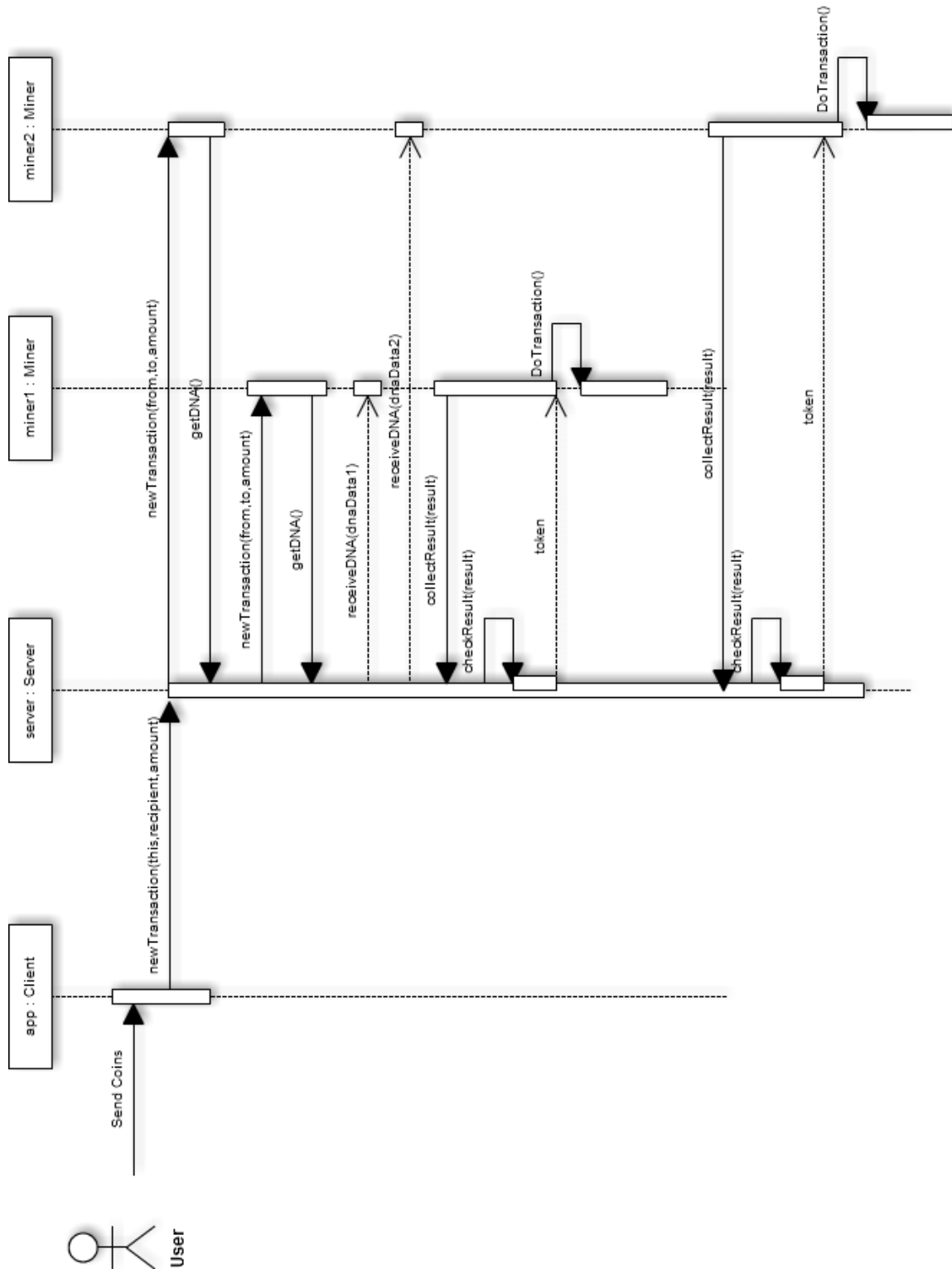
3.4.4 Dynamic Models

3.4.4.1 Sequence Diagrams

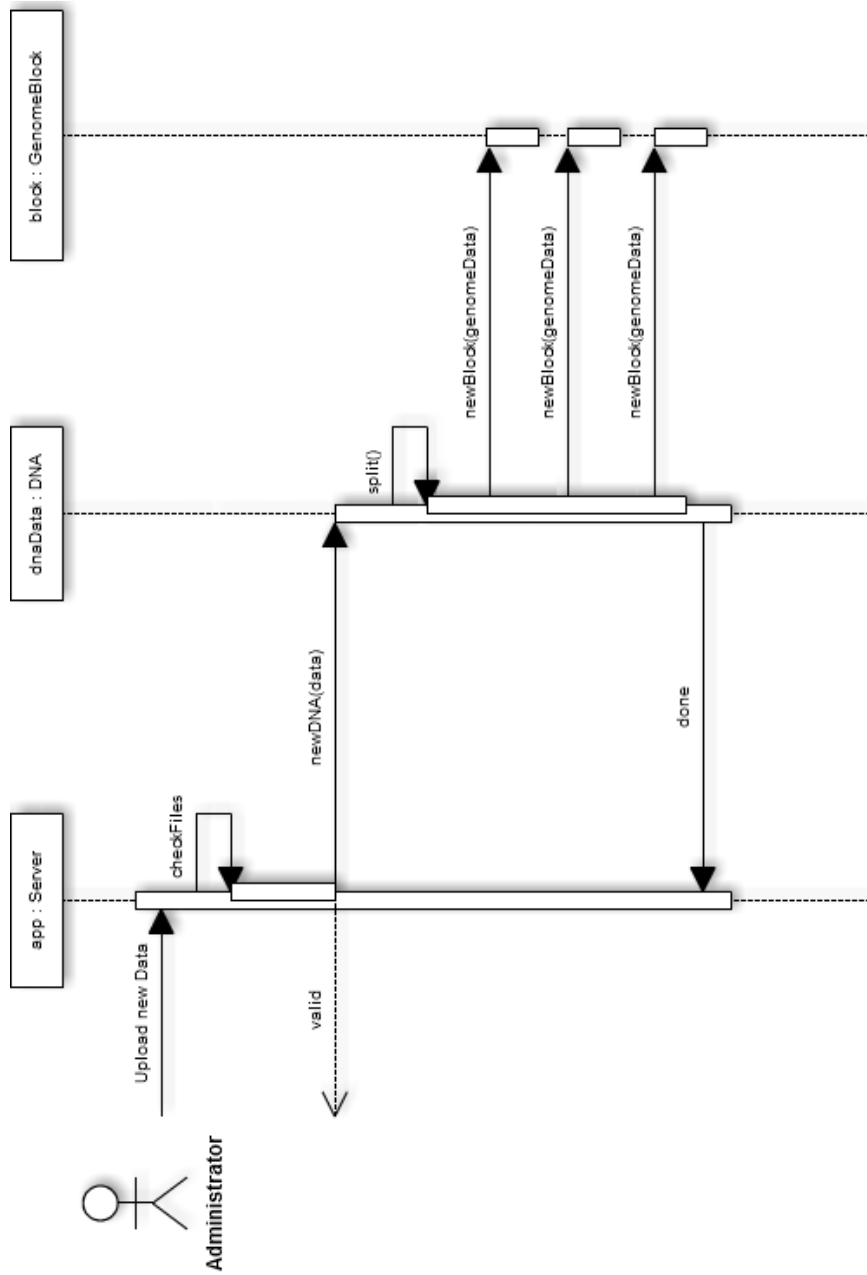
3.4.4.1.1 Creating a Wallet



3.4.4.1.2 Sending Coins

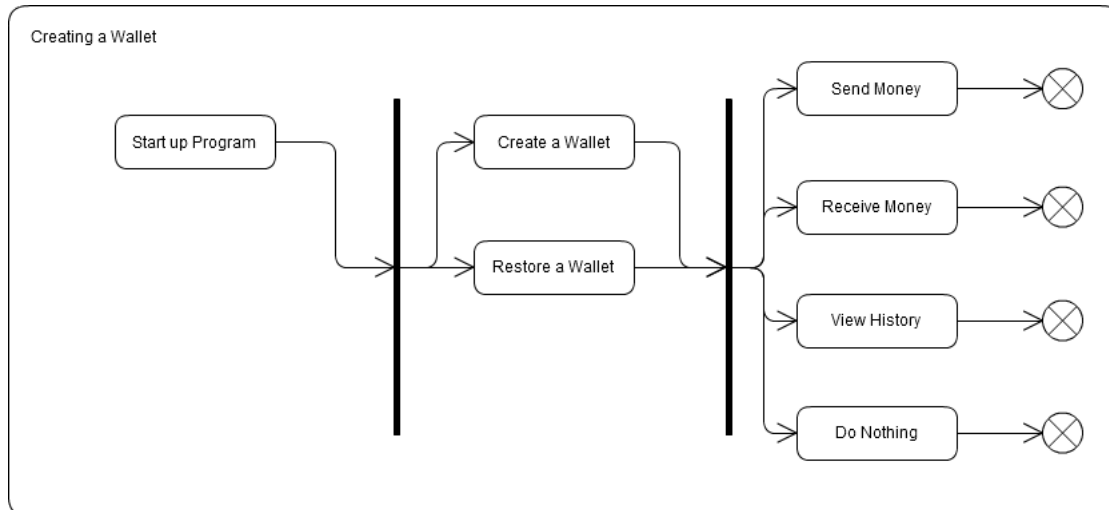


3.4.4.1.3 Uploading new Genome Data

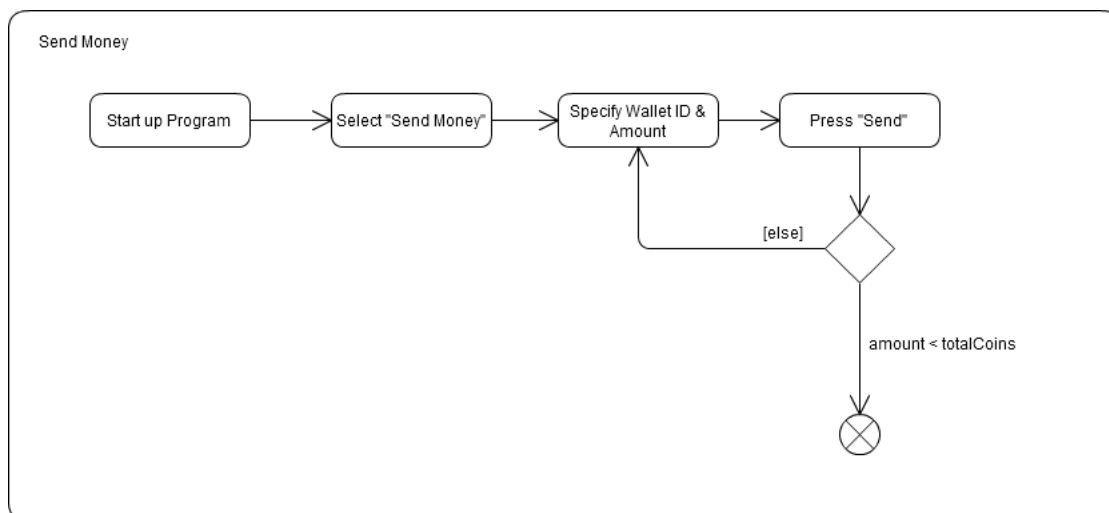


3.4.4.2 Activity Diagrams

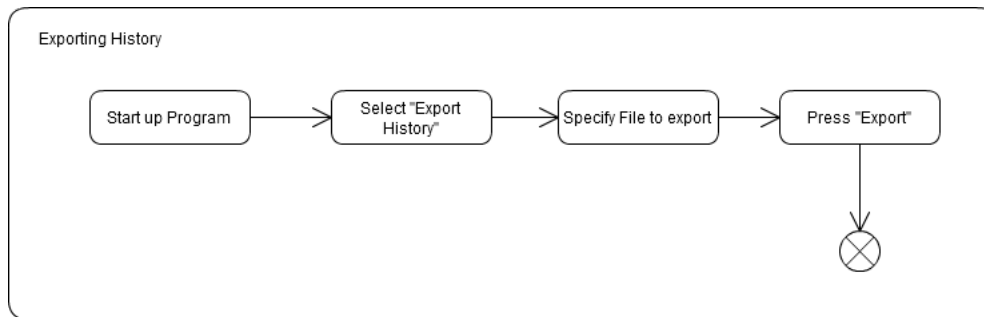
3.4.4.2.1 Creating a Wallet



3.4.4.2.2 Sending Coins

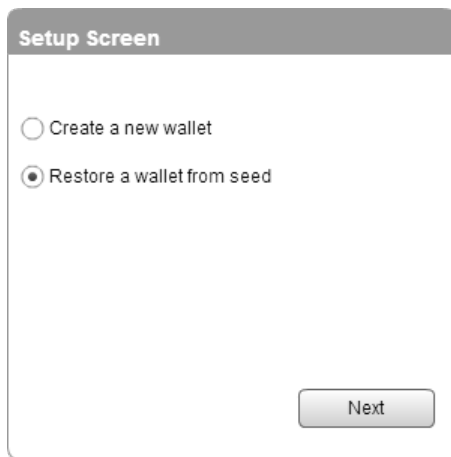


3.4.4.2.3 Exporting Transaction History



3.4.5 User Interface

3.4.5.1 Creating/Restoring a Wallet



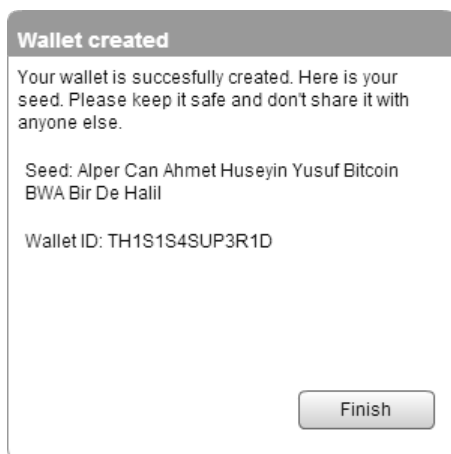
The Setup Screen dialog box has a title bar labeled "Setup Screen". Inside, there are two radio button options: "Create a new wallet" and "Restore a wallet from seed". The "Restore a wallet from seed" option is selected. At the bottom right, there is a "Next" button.

Choosing the first option will proceed to “Create Wallet“ section and the second option will proceed to “Restore Wallet“ section.



The Create Wallet dialog box has a title bar labeled "Create Wallet". Below the title bar, there is a paragraph of text: "Please enter a password for the security of your wallet. You may skip this step if you don't want to secure your wallet." Below this text are two input fields: "Enter password" and "Repeat password". At the bottom, there are two buttons: "Skip" and "Next".

Clicking next or skip will proceed to “Wallet Created“ page. Skipping will leave the wallet unencrypted.



The Wallet created dialog box has a title bar labeled "Wallet created". Below the title bar, there is a paragraph of text: "Your wallet is succesfully created. Here is your seed. Please keep it safe and don't share it with anyone else." Below this text, there is a line of text: "Seed: Alper Can Ahmet Huseyin Yusuf Bitcoin BWA Bir De Halil". Below that, there is another line of text: "Wallet ID: TH1S1S4SUP3R1D". At the bottom right, there is a "Finish" button.

Clicking finish will finish the process and proceed to the coin management client.

Restore Wallet

Please enter the file location of wallet which is going to be restored.

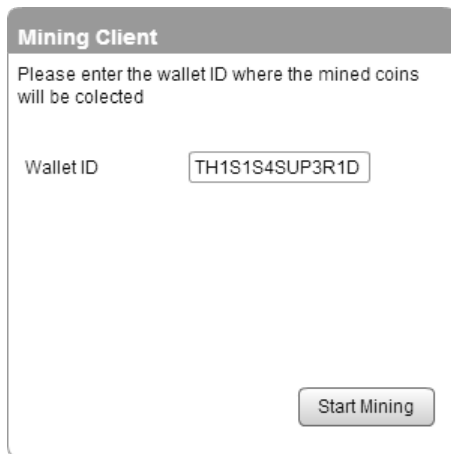
Wallet File:

Choose File

Finish

Clicking finish will finish the process and proceed to the coin management client.

3.4.5.2 Mining Coins



The image shows a window titled "Mining Client". Inside the window, there is a text label "Please enter the wallet ID where the mined coins will be collected". Below this, there is a label "Wallet ID" followed by a text input field containing the value "TH1S1S4SUP3R1D". At the bottom right of the window, there is a button labeled "Start Mining".

Clicking start will minimize the client and start mining.

3.4.5.3 Sending Coins

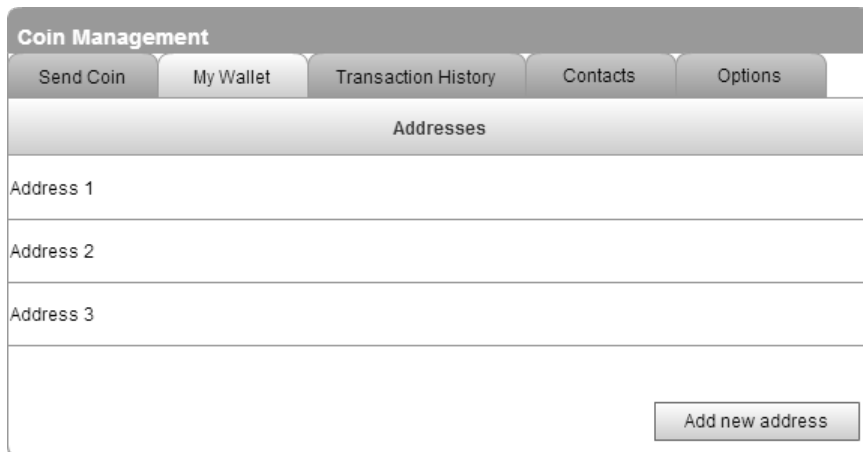


The image shows a 'Coin Management' dialog box with a grey header bar. Below the header is a tabbed interface with four tabs: 'Send Coin', 'Transaction History', 'Contacts', and 'Options'. The 'Send Coin' tab is currently selected. The main area of the dialog contains three input fields: 'Receiver' with a placeholder text 'Wallet ID or Contact name', 'Description', and 'Amount'. A 'Send' button is located in the bottom right corner of the dialog.

Coin Management			
Send Coin	Transaction History	Contacts	Options
Receiver	<input type="text" value="Wallet ID or Contact name"/>		
Description	<input type="text"/>		
Amount	<input type="text"/>		
<div>Send</div>			

Clicking send will send the coins to the specified receiver.

3.4.5.4 Receiving Coins



The screenshot shows a web interface for 'Coin Management'. At the top is a dark grey header bar with the text 'Coin Management' in white. Below this is a row of five tabs: 'Send Coin', 'My Wallet', 'Transaction History', 'Contacts', and 'Options'. The 'My Wallet' tab is currently selected and highlighted. Below the tabs is a section titled 'Addresses' in a light grey bar. Under this section, there are three rows, each labeled 'Address 1', 'Address 2', and 'Address 3' respectively. At the bottom right of the 'Addresses' section, there is a button labeled 'Add new address'.

A wallet can have multiple addresses. This is where the addresses are listed. Add new address button will add a new randomly generated address.

3.4.5.5 Viewing Contacts

Coin Management	
Send Coin	My Wallet
Transaction History	Contacts
Options	
Name	Address
Ahmet	Address 1
Can	Address 2
Alper	Address 3
<div>Add new address</div>	

User can save the contacts here. Add new contact button will lead to a pop-up that is requesting name and address information. User can right-click and delete or modify a contact.

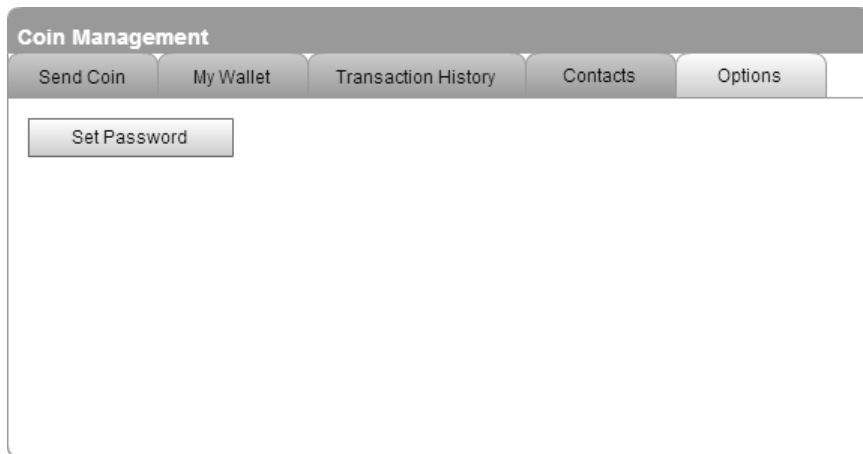
3.4.5.6 Viewing History

Coin Management		
Send Coin	My Wallet	Transaction History
Send Coin	My Wallet	Contacts
Send Coin	My Wallet	Options
Receiver	Description	Amount
Ahmet	Description 1	10
Can	Description 2	-20
Alper	Description 3	40
		<input type="button" value="Export"/>

User can find the transaction history here. This is a read-only view and cannot be modified.

User can export the history using the Export button, which will trigger the file picker for a .csv file to be saved.

3.4.5.7 Options



Set password button will open a dialog that is requesting the old password (if any) and the new desired password.

3.4.5.8 Administration



Add new genome data button will open a file picker. Admin should select a file that contains the genome data for an individual. Then the data will be uploaded to server. Collect results button will download all uncollected results to a folder of choice.

4 References

- [1] <https://bitcoin.org/bitcoin.pdf>
- [2] <https://litecoin.org/>
- [3] <http://peercoin.net/>
- [4] <http://dogecoin.com/>
- [5] <http://blockchain.info/stats>
- [6] <http://boinc.berkeley.edu>