CS 795

Blockchain Technologies

Public Wallet Interface for Ripple

CS 795

Authors:

Madhusudhan, Abhishek Wadki, Suyameendra

May 15, 2017

Contents

1 Abstract

We study the core functionalities that make up a cryptocurrency wallet. Analyzing the different types of wallets available for Ripple, we design and develop a public wallet interface that could be used to send, receive currencies; establish trust links with different accounts. Also, viewing list of transactions made and compatible currencies with associated with the account.

2 Introduction

Ripple

Ripple is a global real time settlement network mainly to address cross border payments. It relies on common shared ledger, which is distributed database storing information about all ripple accounts. The network is "managed by a network of independed validating servers" that constantly compare their transaction records. These servers could belong to anyone including banks or market makers.

Some of the key features of ripple are as follows:

- **Distributed:** No central operator required during direct settlements.
- **Interoperable:** Ability to transact directly on different networks integrating with existing systems and standards.
- **Scalable:** Ability to process the world's cross-bordery payments volume and option to access liquidity through a competitive FX marketplace.
- **Secure:** Transactions are cryptographically signed using ECDSA and ED25519 algorithms. [3]

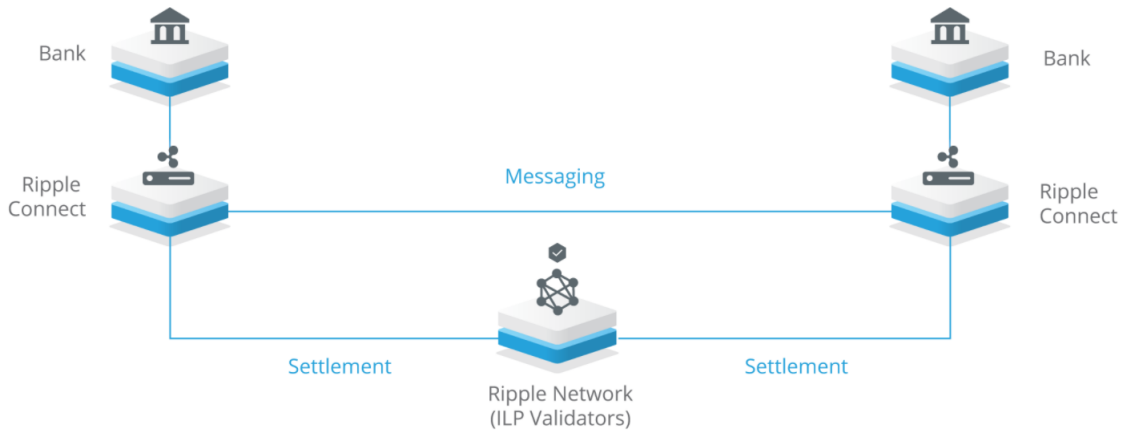


Figure 1: Ripple Network

- Cross border payments require many intermediaries causing settlements to take multiple days creating costs and risks making it difficult to meet market demands.
- Ripple enables cross border payments to execute within seconds providing end to end visibility increasing processing rates thereby lowering operational costs.
- No customer information touches the ripple network.
- Provides ability to trace funds end to end with certainty of delivery minimizing cost of every transaction.

Ripple Consensus Protocol

Ripple's consensus protocol is an asynchronous round-based protocol which is executed by the network's validating servers. At the end of every round, a new last closed ledger is published by all involved

servers. The consensus protocol comprises of three phases: the collection phase, the consensus phase and the ledger closing phase.

In the collection phase, the validating servers collect the transactions that they receive from the network. Upon receiving a transaction, validating servers check its authenticity by verifying the issuer's public key (from the ledger) and they also check the validity of the corresponding signature. Transactions which come equipped with valid signatures are temporarily stored in the candidate set CS for subsequent validation. The validation servers then check the correctness of transactions stored in CS including verifying that enough credit is available in the issuing account by going over the history of all transactions pertaining to that account(in case of an XRP transactions), or the existence of a trust path between the sender and receiver (in case of an IOU payment). Each validating server packages validated transactions in an (authenticated) proposal and broadcasts its proposal in the network. In Ripple this is achieved by constructing a has tree of all validated transactions, and subsequently signing the root tree.

When validating server receives a new proposal from the network, it checks that the proposal's issuer is a server which appears in its UNL and verifies the correctness of the transactions included in the received proposal. Once a transaction reaches 80 % acceptance, it will be removed from the candidate set, checked for double-spending. This transaction will be then appended to the ledger and the balance of the sender/recipient will be appropriately updated. After closing the ledger, transactions which have been received during the consensus phase will be processed, and the next round will start. [5]

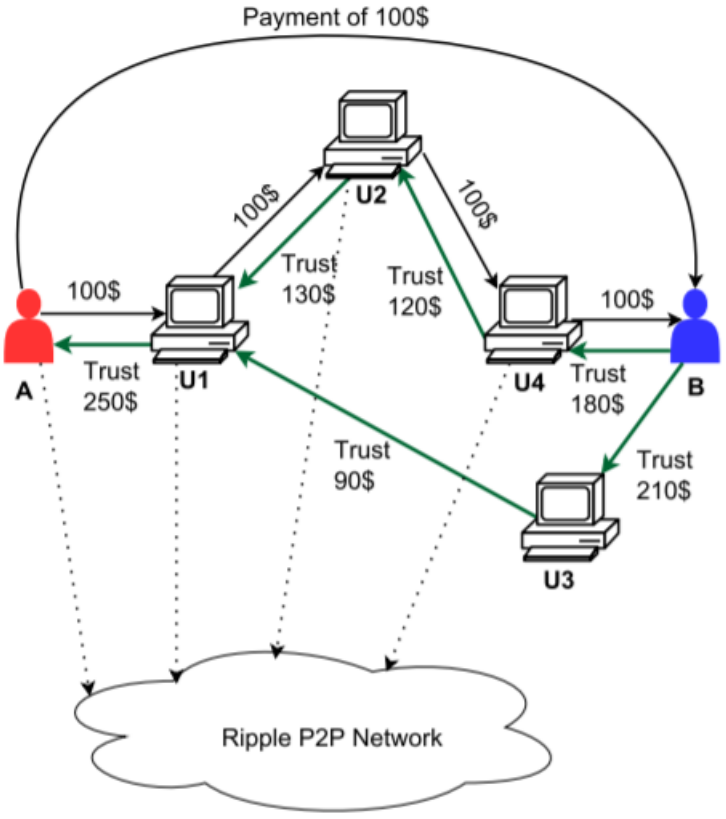


Figure 2: Ripple Network [5]

Cryptocurrency Wallet

A cryptocurrency wallet is a secure digital wallet used to store, send, and receive digital currency. Most coins have an official wallet or a few officially recommended third party wallets. Inorder to use cryptocurrency, you have to hold a cryptocurrency wallet.

There are different types of cryptocurrency wallets as follows:

- **Desktop:** The most common type of wallet wherein an application connects directly to a coin's client.
- **Online:** Are user friendly and the wallet could be accessed from any device over the internet. They are not very secure because private keys are store on another server and are not suitable for holding a large number of cryptocurrencies.[1].
- **Paper:** A QR code printed out for both a public and private key. These hard copies could be stored in a regular wallet. Intended for traders who invest most of their cryptocurrency and not use it as much for purchases.
- **Hardware:** They are stand-alone hardware cold-storage devices that generate keys on the fly while making a transaction. They are secure because of the following reasons:
  1. They have a dedicated hardware that is specifically built to hold cryptocurrency securely like USB devices.
  2. Generate private keys offline.
  3. Could be secured with a password to combat theft. environment.
- **Hybrid:** They address the limitation of an online wallet by encrypting private data before being sent to the online server.

3 Program Design Architecture

Rippld Server

It is a core peer-to-peer server that manages the Ripple Consensus Ledger (RCL). Each rippled server connects to a network of peers, relays cryptographically signed transactions, and maintains a local copy of the complete shared global ledger. The source code for rippled is composed in C++, and is accessible on GitHub under an open-source permit.

Websocket

The WebSocket API uses the WebSocket protocol, available in most browsers and Javascript implementations, to achieve persistent two-way communication. There is not a 1:1 correlation between requests and responses. Some requests prompt the server to send multiple messages back asynchronously; other times, responses may arrive in a different order than the requests that prompted them. The rippled server can be configured to accept secured (wss:), unsecured (ws:) WebSocket connections, or both [2].

Json-RPC

The JSON-RPC API relies on request-response communication via HTTP or HTTPS. (The rippled server can be configured to accept HTTP, HTTPS, or both.) For commands that prompt multiple responses, you can provide a callback URL.

The rippled program can also be used as a quick commandline client to make JSON-RPC requests to a running rippled server. This is only intended for administrative purposes, and is not a supported API.

Any HTTP client like Poster for Firefox or Postman for Chrome could be used to make JSON-RPC calls a rippled server. Most programming languages have a built in library for making HTTP requests [2].

- We chose JSON RPC APIs due to the widespread support for JSON-RPC and availability of standard HTTP library to connect to rippled's JSON-RPC API. We are constructing the request structure at the back-end and then make a JSON-RPC call.

Domain	Port	Notes
s1.ripple.com	51234	General purpose server
s2.ripple.com	51234	Full-history server

Figure 3: Ripple Public Servers

- We Send an HTTP POST request to the root path (/) on the port and IP where the rippled server is listening for JSON-RPC connections. Figure 2 depicts the available public rippled servers and the sample request, response structures.

Error Responses

It is impossible to list all the possible ways an error can occur. Some may occur in the transport layer (for example, loss of network connectivity), in which case the results vary depending on what client and transport you are using. If the rippled server successfully receives your request, it tries to respond in a standardized error format.

4 Functionality

Account Info

- Retrieve information about an account, its activity and its XRP balance.
- All information is relative to a particular version of the ledger [6].

Following are the sample request and response structures:


```
{
    "method": "account_info",
    "params": [
        {
            "account": "rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK",
            "strict": true,
            "ledger_index": "current",
            "queue": true
        }
    ]
}
```


(a) Request


```
{
  "result": {
    "account_data": {
      "Account": "rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK",
      "Balance": "8277639680",
      "Flags": 0,
      "LedgerEntryType": "AccountRoot",
      "OwnerCount": 22,
      "PreviousTxnID": "217016A256C005283E96CC434275B13BD495D6562EB2234EABF0240EABE36DBE",
      "PreviousTxnLgrSeq": 1171576,
      "Sequence": 69,
      "index": "769E6AC6B48A445A7727D7ED4676B748F3FFF59E10D3063C77E1B81A1AB1DCF3"
    },
    "ledger_current_index": 1178883,
    "queue_data": {
      "txn_count": 0
    },
    "status": "success",
    "validated": false
  }
}
```


(b) Response

Figure 4: Sample JSON Structures

- Following details are retrieved as shown in figure 4:
    1. Account Number: Denotes the address of the wallet.
    2. Balance: The total number of XRP balance held by the account currently.
    3. Previous Transaction ID: Hash of the previous transaction.

Transaction History

- Retrieves a list of transactions that involved the specified account.
- At present, we have implemented two out of six supported ripple transactions which are as follows:
  1. Payment: A payment transaction represents a transfer of value from one account to another. Depending on the path taken, additional exchanges of value may occur atomically to facilitate payment.
  2. TrustSet: A type of transaction used to create or modify a trust line between two accounts [4].

We retrieve and display both the types of transactions to the user.

Send Money

- Send a transaction of type "Payment" to the network to be confirmed and included in future ledgers.
- Requires Sender Address, Receiver address, Sender's Secret key, Amount and the Currency in which the money is to be sent.
- There are only two possible responses for a "transaction" i.e. success or failure. Both these cases are handled by our application [4].

Establish Trust Links

- This functionality is posted on the ledger with a transaction type of "TrustSet".
- In order to receive payments in anything other than XRP you need to extend "trust" to some other Ripple account. This specifies that you are willing to have other account "hold" your payment for you for later redemption.
- You can set the maximum amount of trust whereas a bank account will not let you place an upper limit on your bank balance.

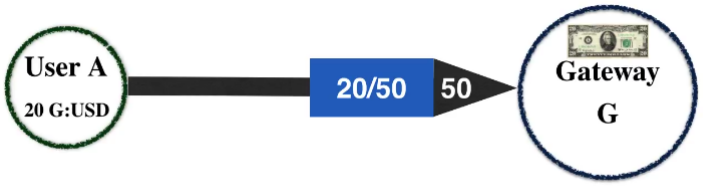


Figure 5: Trust link example

- In the example above, the User "A" trusts gateway "G" with 50 USD. If User "A" is going to give Gateway G 20 dollars, so here 20/50 belongs to user "A" represented as "20 G: USD".[11]

Listing Compatible Currencies

Retrieving a list of compatible currencies for a specific user. Lists compatible currencies while performing the following operations:

1. Send: All that currencies that can be sent by the user.
2. Receive: All the currencies that can be received by the user [6].

Listing Account Trust Lines

- Retrieves information about the account's lines of trust, including balances in all non-XRP currencies, assets.
- All information retrieved is relative to a particular version of the ledger.

Following information are retrieved:

1. Account: Address of the account with which a trust is established.
2. Limit: The maximum amount of currency that the counterparty is willing to owe the perspective account.
3. Quality In/Quality Out: Rate at which the account values incoming and outgoing balances. 0 is treated as 1:1 ration.
4. Currency: A currency code identifying what currency this trust line could hold.
5. No Ripple Peer: True if the user doesn't want to allow rippling, false otherwise [6].

5 Preview

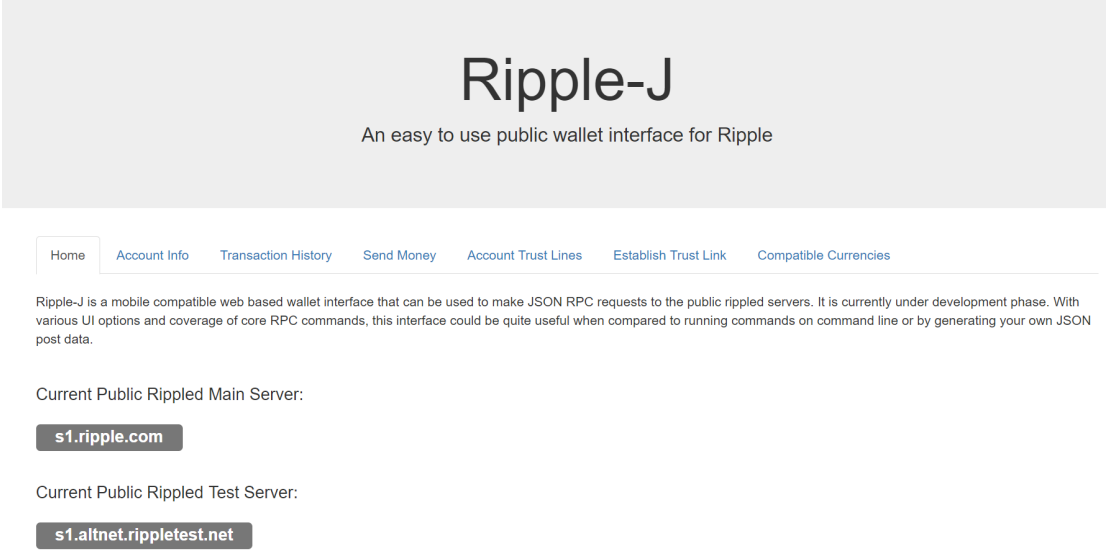


Figure 6: Home Page

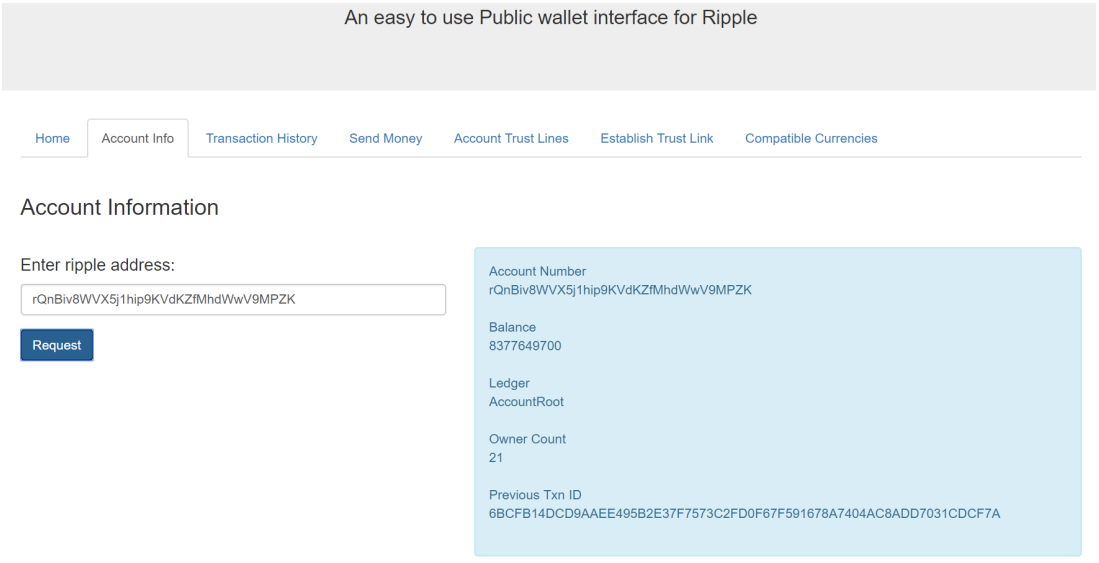


Figure 7: Account Info

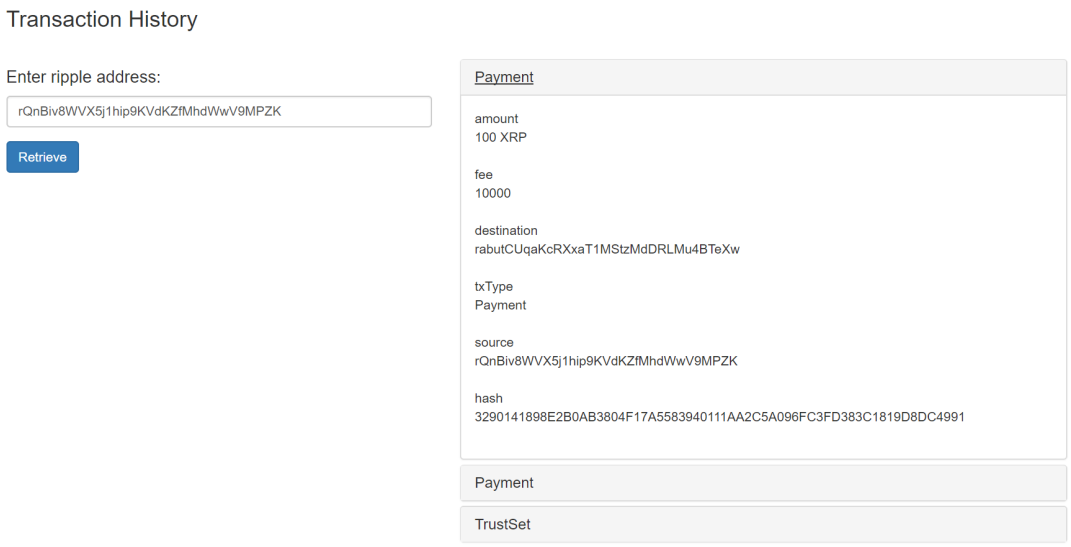
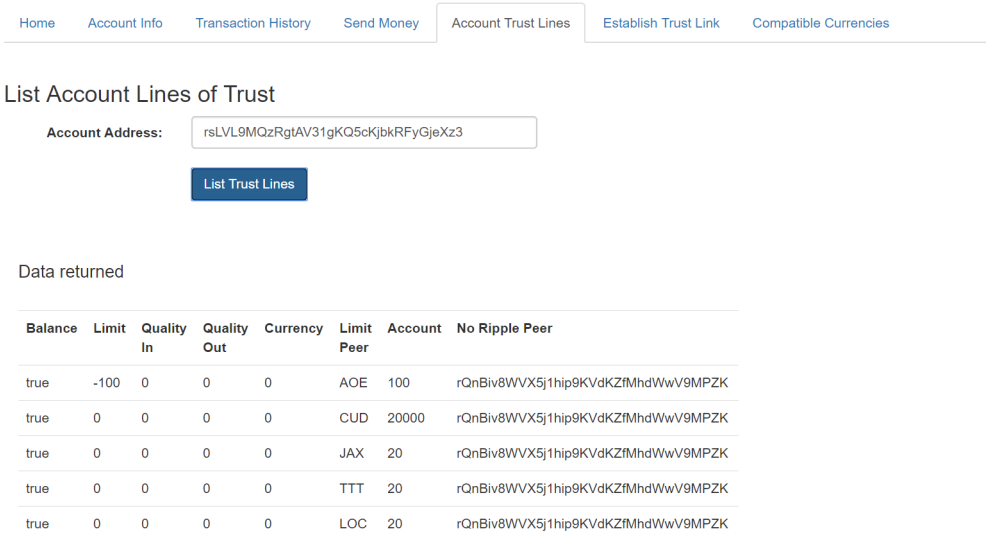


Figure 8: Transaction History



Balance	Limit	Quality In	Quality Out	Currency	Limit Peer	Account	No Ripple Peer
true	-100	0	0	0	AOE	100	rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK
true	0	0	0	0	CUD	20000	rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK
true	0	0	0	0	JAX	20	rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK
true	0	0	0	0	TTT	20	rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK
true	0	0	0	0	LOC	20	rQnBiv8WVX5j1hip9KVdKZfMhdWwV9MPZK

Figure 9: Lines of Trust

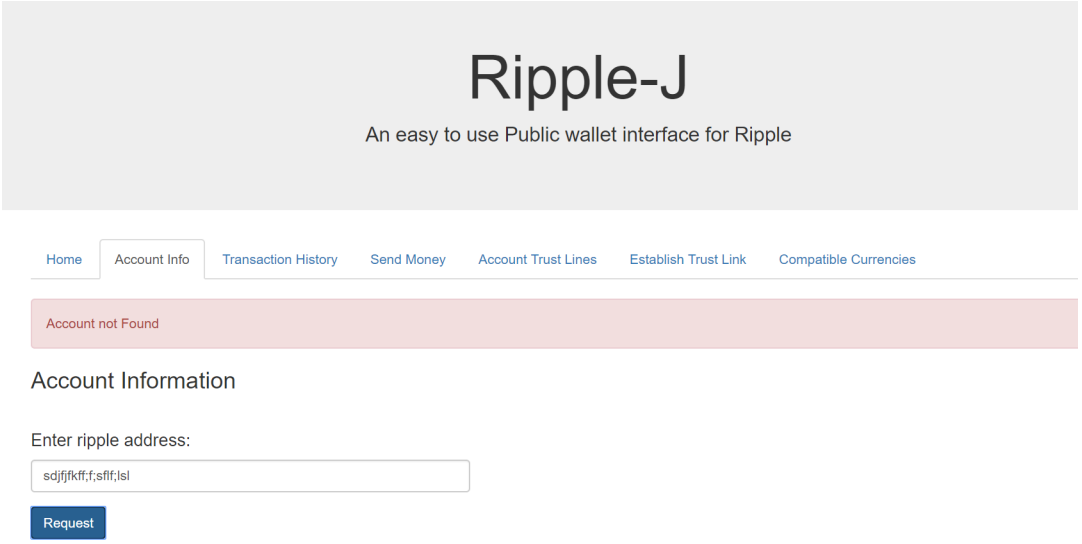


Figure 10: Sample Error

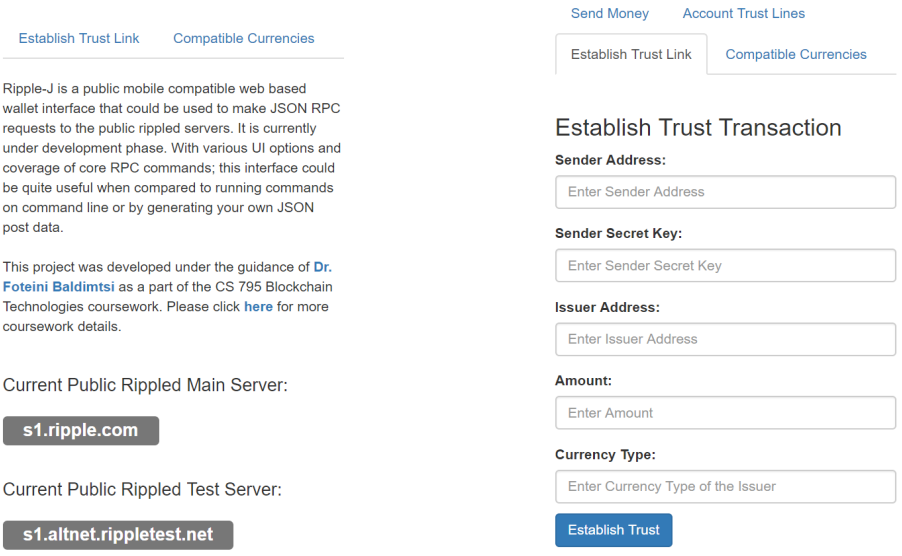


Figure 11: Mobile Compatibility

6 In-comparison with other wallets

Web Wallet - Gatehub [7]

The Only web wallet application available. It enables you to manage assets on multiple wallets, send and receive payments via ripple.

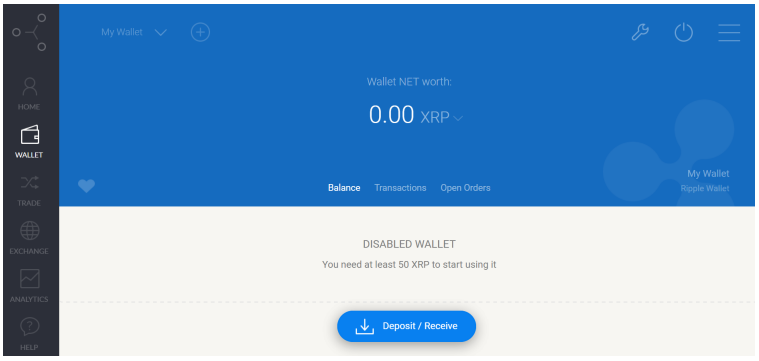


Figure 12: Gatehub Wallet

Pros:

- Apart from the core functionality, it has some very interesting features like Analytics and Exchange of money between the currencies. These functionalities could be integrated into our application as well.

Cons:

- Lack of Mobile compatibility
- Not official

Mobile Wallet - Instant Ripple Wallet [8]

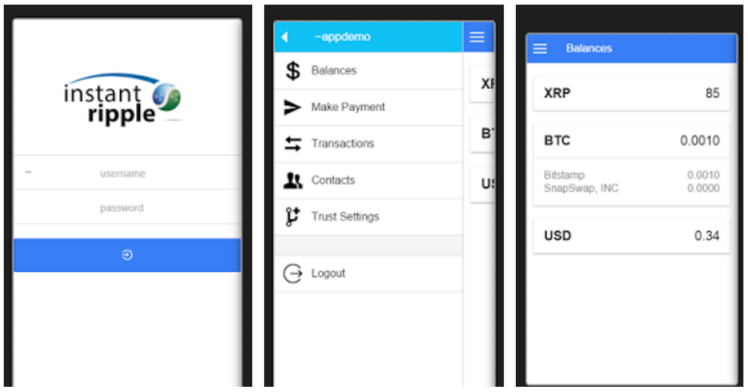


Figure 13: Gatehub Wallet

Pros:

- First attempt. Gave an idea of a wallet application.

- Basic functionality

Cons:

- Functionalities do not work and hence taken down by app store
- Not all core functionalities were implemented.

Ripple Client for IOS

A mobile application for Ripple released only on iOS by ripple labs [9].

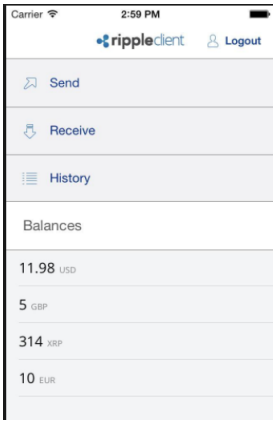


Figure 14: Gatehub Wallet

Pros:

- First decent mobile wallet application for Ripple.
- Includes core functionalities.

Cons:

- Lacks additional functionalities
- Not available on all platforms

7 Future Work

The current version of the project is made public and has been hosted using Amazon AWS Cloud EC 2 instance.

1. http://52.39.10.40/RippleJsonRPC2/
2. http://54.70.214.45:8080/RippleJsonRPC2/

We aim to continue further development on this project by enhancing and adding functionalities like secure storage of keys, viewing real time exchange rates and many more that are exclusive to ripple.

We also plan to add interesting features like Analytics and Exchange of money between the currencies.In addition to this, we also have a mobile wallet application for android in development.

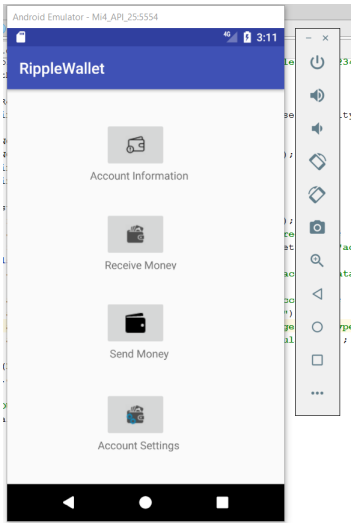


Figure 15: Mobile Wallet

8 Technologies Used

User Interface: HTML, BootStrap (Mobile Compatibility) [10], JQuery

Backend: J2EE, Servlets, AWS EC2 [12].

9 Conclusion

Ripple is a very interesting and fast growing credit network protocol and hence there is a substantial need for a simple and efficient wallet application that can handle all the core functions of Ripple. Apart from Gatehub, there are no wallets available in the market today. Hence, we have developed Ripple-J, a web wallet application for ripple that handles these core functions as well as provides a mobile compatibility for cell phone users, as currently there are no good mobile wallets available. We also plan to package our wallet application into a Mobile app, so that the users can easily access their accounts and send money instantly just like any other banking wallet.

References

[1] http://cryptocurrencyfacts.com/what-is-a-cryptocurrency-wallet/.

[2] Ripple Labs Inc. JSON WebSocket https://ripple.com/build/rippled-apis/.

[3] Ripple Labs Inc. Key Features. https://ripple.com/technology/

[4] Ripple Labs Inc. Transactions. https://ripple.com/build/transactions/

[5] Ripple: Overview and Outlook Frederik Armknecht1, Ghassan O. Karame2, Avikarsha Mandal3, Franck Youssef2 and Erik Zenner3

[6] Ripple API's: https://ripple.com/build/rippled-apis/

[7] Gatehub: https://www.gatehub.net/

[8] Instant Ripple Wallet: https://play.google.com/store/apps/

[9] Ripple Client for IOS: http://rippletalk.blogspot.com/2014/02/trading-xrp-client-from-ripple-labs.html

[10] BootStrap: https://www.w3schools.com/bootstrap/

[11] Trust Link Example: https://www.youtube.com/watch?v=w-rv55TIxgk.

[12] AWS EC2: https://aws.amazon.com/ec2