Expanding on Multiple Cryptocurrency and Wallet Software

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Nikita Saxena

Fall, 2021

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Technical Writing Advisor: Rosanne Vrugtman PhD, Department of Computer Science

Technical Advisor: Daniel G. Graham PhD, Department of Computer Science

Expanding on Multiple Cryptocurrency and Wallet Software

CS 4991 Capstone Report, 2021

Nikita Saxena
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville Virginia USA
ns5ub@virginia.edu

ABSTRACT

The vast array of cryptocurrencies and wallets available create a problem of scale for any software designed to interact with multiple types. Each potential currency or wallet provides its own unique challenges due to the lack of uniformity in implementation and tools. A technical lead designed a framework to simplify interaction with multiple currencies via abstraction. I worked as an intern along with a team of full hires to understand the established codebase and framework, fix potential bugs, and research the tools necessary to implement the cryptocurrency Monero within the framework.

The framework generalized the interaction into stages that are present across several cryptocurrencies. The project handled the minutiae of each currency and its corresponding wallet individually, so users only had to know their own currency and wallet. As such, bug fixes and research into additional currencies helped expand the potential array of currencies with which the framework, and thus the end user, can interact with.

1 Introduction

There are over 13,000 cryptocurrencies in existence. New types of cryptocurrencies are regularly created, each with its own use case and corresponding design quirks. [6] Many cryptocurrencies utilize blockchain technology to allow transactions without a third-party mediator. The industry of financial cryptocurrency is expected to grow in the coming years, especially as more large institutions such as PayPal and Amazon are making moves to accept cryptocurrency payments. [10] Granting Consumers greater flexibility in their cryptocurrency and wallet of type is essential to this growth, but can present a software development challenge due to the variety inherent in cryptocurrencies.

2 Background

Despite the large number of currencies, 70% of the market in cryptocurrencies is made up of the top 5 coins. There is a subset of crypto-assets that are “payment-focused” coins including Bitcoin, Litecoin, and Ethereum. [6] These are thus the most relevant cryptocurrencies for a transaction-oriented software to be able to handle. It also implies similarities in use case, and thus, potentially transaction functionality exposed for the coins and their corresponding wallets. This does not imply a uniform implementation however. [1]

Cryptocurrencies utilize blockchain technology to perform transactions without the need for a third-party authority such as a bank. A blockchain is a type of distributed ledger and each user has their own copy of the ledger known as a node. Each node must run the relevant blockchain client software in order to exchange assets. To confirm a transaction on the ledger, all nodes must agree on the transaction, a process that can often require a significant amount of time to complete, and must be accounted for in software. [4] Different types of cryptocurrencies are deployed on different blockchains, even using variants of the technology. Transactions using a currency can be made on the network using the appropriate blockchain client. In a multicurrency software this implies the use of multiple clients. [1]

Consumers can participate in the blockchain and make transactions with a digital wallet. Digital wallets are usually designed to store a specific type of coin. Each wallet has a public key that acts as an address for transactions to be routed to and a corresponding private key. The private key is not shared, but is required to approve any transfer of cryptocurrency out of the wallet. [1] Wallets usually have functionality to store, send, and receive a specific type of cryptocurrency. In order to interact with multiple types of currencies, a software must be able to interact with a multitude of wallets.

3 Related Work

Multi-wallet software is directly related to this project. Software such as “FreeWallet” [13], “Exodus” [7], and “Coinomi” [5] provide custom individual wallets for each cryptocurrency they are able to work with (Coinomi Ltd., n.d; Exodus n.d.; Wallet Services Ltd., n.d.). In contrast, this multi-wallet framework works with external wallets of the user’s choice. An alternate design for working with multiple currencies is multi-asset wallets like “ViaWallet” which provides a single wallet that stores multiple cryptocurrencies. ViaWallet also exposes some of its blockchain functionality in an API for public use, demonstrating similar developer-oriented design goals. [12] Previous research

has explored new methodology for multi-asset system design. Tian et al. [8] investigated an alternative transaction scheme for cross-cryptocurrency transaction using smart contracts. Xu et al. [9] designed a system to maintain privacy and security when managing multi-asset cryptocurrency transactions.

4 Project Design

I worked on a system designed by a technical lead to solve the challenges of multi-currency software. The system abstracts away the minutiae of a transaction by splitting the process into two portions: connecting to the daemon and interacting with the software wallet. The developer implements a standardized set of functionalities for each new wallet/cryptocurrency while the user must only know their own currency and wallet. I was initially tasked with exploring and understanding this system so that I could fix bugs and assist in implementing new cryptocurrencies. I then researched whether the publicly available resources for the coin Monero would allow implementation within the multi-wallet framework as is.

A technical lead built a prototype software to implement and test project features before deployment. I was directed to use test-driven development since functionality for each currency was predefined by the framework. Test Networks were used along with replica currency to test functionality without the risk of using real currency. Project features were split across developers by currencies and/or wallet as each implementation was independent. Planning and general development was handled with a loose agile development style with weekly sprints.

4.1 Framework: Daemon

In order to complete a transaction software must interact with the blockchain to confirm the new transfer, adding it to the "public record" so to speak. A daemon is a program running in the background, and in the case of cryptocurrency, connects a given wallet to the appropriate blockchain. This process requires both the software to have at least a partial record of the current blockchain and the ability to queue up the transaction to be validated via mining.

The program handling the daemon for a given cryptocurrency encapsulated the logic for starting, stopping, and checking if the daemon was still running amongst other functionality. This facilitates switching between cryptocurrencies as each is guaranteed to have the exposed functionality to set up, reset, and tear down its daemon. This encapsulation also included the ability to send specific commands to the daemon. Interacting with a daemon is largely asynchronous since the startup, tear down, and mining each take a variable amount of time to complete.

In order to work with the daemon and the larger blockchain, I also had to investigate whether a potential "Test Network" existed for a given coin. This is a network that follows the same rules as the real blockchain, but works with dummy versions of the currency with no real value. This was essential to the testing phase of the application as working with real money was impractical and potentially expensive in the event of bugged code. The ability to create a Private Regression Test Network, a completely private version of the mentioned Test Network, was also desirable. A private test network allows users to manipulate the conditions of the underlying blockchain and generate fake coins on the fly without connecting to external network and services to acquire coins on the Test Network.

4.2 Framework: Software Wallet

In order to complete a transaction, the program must interact with the user-loaded software wallets to transfer currencies to and from. Software wallets are variable in how they store the relevant public and private keys, their intended functionality in transfers, and protocols. This portion of the framework encapsulates the implementation of the actual transactions, getting the relevant keys from the user's wallet and performing appropriate transfers. Once again, the program should implement some standard set of functions, exposing their functionality to the user and greater software.

This portion required research into the appropriate Remote Procedure Call (RPC) for the given wallet. This is the system by which the software wallet connects to an already running instance of the Daemon. The program uses the RPC in order to actually send any given transaction to the daemon for approval. Many software wallets/cryptocurrencies have an API for a Wallet RPC that already has many basic transaction functionalities implemented, in which case the developer is largely writing a wrapper around them. For example, the Bitcoin API has the ability to load in a wallet, get balances, and transfer coins to an address amongst other things. [3] This process, similar to connecting to the Daemon, isn't instantaneous.

4.3 Understanding the Framework: Ethereum

Once I had a grasp on the framework, I was assigned to fix some issues with the Ethereum implementation. To understand the framework, I studied the given codebase as well as the sample implementations for Bitcoin and the partial implementation for Ethereum. My understanding of the framework is discussed in 4.1 and 4.2. I also performed general research and worked with other team members to gain domain knowledge about cryptocurrency in order to understand why the code was designed as it was.

From there I investigated the implementation for the Ethereum implementation and identified issues causing the code to fail. In particular I studied asynchronous programming concepts in resolving issues of timing within the program. For example, even while using a particular wallet's API, Ethereum in this case, maintaining the node can require a variable amount of time depending of software specifications. As such, the start-up and shut down process of different instances of daemons were leaving behind artifacts due to the timing of the clean-up functionality. I also analyzed program output to identify a faulty genesis block in a private test network, pushed a working version based on additional research.

4.2 New Coin: Monero

I then researched a possible implementation of Monero in the multi-wallet framework. Monero is a privacy-focused coin as opposed to a transaction-oriented coin like the other implemented currencies (Bitcoin, Ethereum, and Litecoin) causing it to have unique quirks in its management of its public and private keys. Even so, Monero has robust documentation of its features and is one of the more popular privacy-focused coins used in transactions, making it a viable option to implement within the framework. [2]

Following the format of the framework, I first visited the official Monero resources to research its daemon. The official website, getmonero.org had both a daemon and associated command line tools, as well as resources to verify those downloads. In addition, they have documentation for a Test Network and faucet to acquire fake Monero to perform tests with. If a Private Test Network is preferred, some modifications have to be made to work with the RPC. [11]

While the framework was created with an assumption of just a private and public key, Monero Wallets have associated “Spend” and “View” keys, of which there are private and public variants. In addition, wallets are required to have an associated “Seed Phrase” that allows the owner to regain their wallet. As such, there are more files to keep track of compared to the transaction-focused currencies.

In addition, Monero has both a Wallet RPC and a Daemon RPC. The former for interacting with the wallet, and the latter for allowing the wallet to interact with the blockchain. This avoids the Daemon RPC having access to the private keys in the wallet. Despite this difference, the Wallet RPC contains similar commands to the transaction currencies including the ability to load a wallet and the ability to start transfers to an address. I also researched open-source libraries that are able put wrappers around the official RPC to wrap together operations involving the two types and use languages such as Javascript. [11]

5 Results

The proposed framework successfully encapsulated the important elements of a general cryptocurrency transaction based on its ability to successfully handle the major transaction cryptocurrencies. I helped expand the potential array of currencies the framework can handle by investigating issues with the Ethereum implementation and researching how Monero, a different type of coin, could fit into the framework. Despite having key differences in file system and RPC, Monero can be implemented using the same general framework.

6 Conclusion

The proposed framework allows multi-cryptocurrency software to interact with a uniform set of functionalities to perform transactions rather than having to deal with the minutiae of each individual cryptocurrency. The encapsulated start up and tear down also allows the software to switch between cryptocurrencies if required. It also allows developers to divide work efficiently as each currency can be implemented independently. The framework is also robust and can interact with privacy-focused cryptocurrencies such as Monero in addition to the transaction-focused.

7 Future Work

To expand on this framework, additional alternative cryptocurrencies could be researched and implemented within the system. This could be done on the basis of the most relevant cryptocurrencies at the time based on transaction metrics. The framework itself could also be expanded in the types of default functionality it exposes in terms of types of addresses it can handle for transactions. This should also be accompanied by research into the most relevant functionalities to implement within the system.

8 UVA Program Evaluation

The University of Virginia CS Program sufficiently prepared me to participate in this project. In particular, the course “CS 3240 Advanced Software Development Techniques” was essential in introducing agile development, gaining experience with source control such as Git, and in using Test-Driven Development. The software design principles introduced in this class such as MVC as well as the general principles of abstraction and encapsulation in “CS 2110 Introduction to Software Development” were essential to quickly getting up to speed on this project.

REFERENCES

[1] Andria Van der Merwe. 2021. A Taxonomy of Cryptocurrencies and Other Digital Assets. Review of Business 41, 1 (January 2021), 30–43.

[2] Anon. Unofficial monero documentation. Retrieved November 2, 2021 from https://monerodocs.org/

[3] Bitcoin Project. Reference. Retrieved November 2, 2021 from https://developer.bitcoin.org/reference/

[4] Bob Tarzey. 2019. Inside Blockchain and its various applications. (September 2019). Retrieved November 2, 2021 from https://www.computerweekly.com/feature/Inside-blockchain-and-its-various-applications

[5] Coinomi Ltd. The blockchain wallet trusted by millions. Retrieved November 2, 2021 from https://www.coinomi.com/en/#features

[6] Connor Sephton. How many cryptocurrencies are there? Retrieved November 2, 2021 from https://currency.com/how-many-cryptocurrencies-are-there

[7] Exodus Movement Inc. Download Exodus. Retrieved November 2, 2021 from https://www.exodus.com/desktop/

[8] Hangyu Tian et al. 2021. Enabling Cross-Chain Transactions: A decentralized cryptocurrency exchange protocol. IEEE Transactions on Information Forensics and Security 16 (2021), 3928–3941. DOI:http://dx.doi.org/10.1109/tifs.2021.3096124

[9] Lei Xu et al. 2020. PrivateEx. Proceedings of the 35th Annual ACM Symposium on Applied Computing (2020). DOI:http://dx.doi.org/10.1145/3341105.3373901

[10] Ryan Harr. 2021. Future of cryptocurrency in 2021 and beyond . (October 2021). Retrieved November 2, 2021 from https://time.com/nextadvisor/investing/cryptocurrency/future-of-cryptocurrency/

[11] The Monero Project. Downloads. Retrieved November 2, 2021 from https://www.getmonero.org/downloads/

[12] ViaWallet. Multi-chain & multi-cryptocurrency wallet. Retrieved November 2, 2021 from https://viawallet.com/?lang=en_US
[13] Wallet Services Ltd. Supported cryptocurrencies and erc20 tokens in Wallet for IOS, Android and desktop. Retrieved November 2, 2021 from https://freewallet.org/assets
.