FROSTDAO: Collective Ownership of wealth using FROST

Rahim Klabér

July 10, 2023

Abstract

The internet has revolutionized how individuals collaborate and work towards a common goal. Individuals from different countries can form informal organizations. The individuals can be thought of as owners of the organization. Thus, they should have a say in managing the funds. However, the organizations' informality makes using banks challenging. This thesis presents a peer-2-peer system on top of Bitcoin and threshold signatures, allowing for collective wealth management. The system allows anyone to create Bitcoin wallets jointly controlled by a group using a threshold-signature scheme. The system is implemented as an Android application and relies on no central party, allowing the system to be used by anyone worldwide. Our expirements show that our system is practical for real-world use. Joining an organization in the system and creating a transaction to spend funds can be done in under a minute.

1 Introduction

In their simplest form, banks act as a middleman between depositors and borrowers. They pool the deposits together and give out personal and corporate loans taking profit by charging a fee. In the years leading to the 2008 financial crisis, banks invested in excessively risky loans using depositors' funds, requiring government bailouts [1]. Recently, the financial system has once again been put to the test with the failures of multiple banks [2]. Banks act as gateways to today's financial world. Without a bank, a person cannot easily invest, pay online or get loans. Nevertheless, these bailouts have shown that bankers may recklessly act in the disinterest of their customers for profit.

Bitcoin presents itself as an alternative to the global financial system [3]. It gave individuals an alternative to banks by allowing anyone to transact on the Internet securely. With Bitcoin, no banks can invest your money into risky products. Some thought Bitcoin could be an alternative to the current financial system [4]. However, Bitcoin has primarily remained a tool for speculation [5]. High fees, low scalability, and a lack of user-friendly applications hinder Bitcoin use [6].

Collective wealth management would allow a group of individuals to manage the group's funds collectively. Each individual has an equal say in how the funds are used. Collective wealth management is a crucial first step to an alternative financial system. Such a system would allow a group of individuals to be their own bank, where each individual is part owner. Collective wealth management allows money to be pooled and invested. This process is transparent and a majority of the participants need to agree for any action to be taken. The idea is similar to Decentralized Autonomous Organizations (DAOs) on other Blockchains, like Ethereum [7], [8]. DAOs are Blockchain-based organizations operating autonomously without central control by using smart contracts. While collective ownership of Bitcoins is possible, it is impractical due to high fees and low scalability [9].

In this thesis, we contribute to the goal of making Bitcoin an alternative to the financial system. We describe and partially implement a critical primitive for the collective ownership of wealth. Using this primitive, individuals can create shared Bitcoin accounts with hundreds of others. Anyone can use our system, which is compatible with existing Bitcoin tools and services.

Specifically, this thesis makes the following contributions:

- *Collective wealth* - We designed a system allowing for the collective ownership of Bitcoins and implemented it as an Android application.
- Performance analysis - We analyze the scalability of our system using various experiments. As part of this, we uncovered performance issues with IPv8's EVA protocol.

2 Problem Description

The internet has revolutionized how individuals collaborate and work towards a common goal [10], [11]. Individuals working together over the internet form informal organizations. One example is Wikipedia, a free encyclopedia with thousands of contributors [12]. The contributors can be considered owners of the organization, as without them, the organization would not exist anymore. As such, the contributors should have a say in managing the organization's funds. However, Collectively managing wealth in such organizations is challenging. Traditionally, a company would be established, allowing for the creation of a business bank account. However, Establishing a company can be complicated and cumbersome, particularly when the individuals involved are from different countries. Additionally, contributors can easily join or leave the organization. Companies are incompatible with this idea as there is high overhead when joining or leaving a company. Existing financial services do not address these issues, highlighting the need for a novel solution.

People are already familiar with banks. Therefore, in order to be a viable option, any solution to the problem must be competitive with banks. In particular, easy-of-use is essential, as anyone should be able to use the solution. Relying on a central actor could result in access being withheld from certain people or certain countries. Thus, the solution should not rely on any central actor. The solution should be scalable, as an organization might have hundreds or even thousands of members.

One potential solution is Decentralized Autonomous Organizations (DAOs) on smart contract enabled Blockchains [8]. DAOs use smart contracts to enable groups of users to control an account. There are various mature DAO tools available. However, DAOs have a tendency to centralize power in a small group [13]. In addition, smart contracts are complex, which has resulted in many hacks [14].

3 System Design

We aim to solve the problem by creating a peer-2-peer, leaderless, and decentralized system. This system will allow groups of individuals to form organizations similar to Decentralized Autonomous Organizations (DAOs) [8] that enable them to collectively and democratically manage their wealth.

3.1 Bitcoin threshold signatures

The cornerstone of our system is the combination of Bitcoin and the FROST threshold signature scheme [15]. Bitcoin allows anyone to send and receive money over the Internet. Bitcoin is decentralized and has tamper-proof and verifiable transactions, allowing anyone to use it without relying on a central party. In our system, each group of participants jointly controls a Bitcoin account that requires a majority to spend funds. We use the FROST threshold-signature scheme [15] to enable collective wealth management without using complex smart contracts. A threshold-signature scheme allows t members of a group of size n, where $t \leq n$, to create a signature jointly. Compared to traditional collective management on Bitcoin [16], threshold signatures can scale to hundreds of participants. In addition, Bitcoin accounts that use threshold signatures are indistinguishable from standard accounts, allowing our design to be compatible with existing tools.

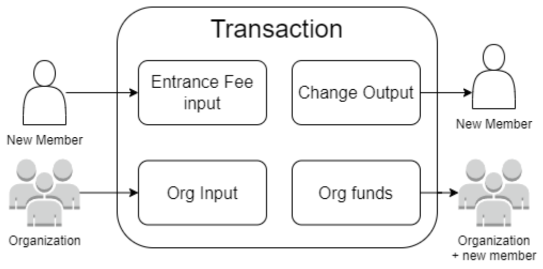

Figure 1: Bitcoin transaction for joining an organization when an entrance fee is needed.

Threshold signatures have a significant drawback. Joining or leaving a group requires every participant to be online. A single participant could temporarily halt the joining process by choosing not to participate. To counteract this, a maximum inactivity duration can be specified, after which the offending member is kicked.

3.2 Decentralized Communication

To support our goal of decentralization, the system uses a peer-2-peer network for communication. Each participant will communicate with another directly to prevent reliance on any central actor. We rely on the IPv8 library for communication [17]. IPv8 allows for the construction of fully peer-2-peer networks through so-called *Communities*. *Communities* are peer-2-peer networks that contain application-specific functionality. Additionally, IPv8 employs hole-punching, which allows devices to communicate, even if they do not have a dedicated public IP address [18].

3.3 Governance

Our governance module sits on top of IPv8's networking and enables democratic decision-making for every action taken by the group. The module is responsible for two things: handling join requests and handling proposals. Join requests allow individuals to request membership, while proposals enable members to suggest actions for the group to take.

Adding a new member to the group involves creating a new Bitcoin threshold account and transferring funds from the old account to the new account. The process has four messages, depicted in Figure 3. It consists of the following steps:

1. The new member creates and broadcasts a request to join the group with the *join request* message.

2. The group members receive the request and send back responses with their votes using the *join request response* message. The message contains a Boolean value that represents agreement or disagreement. The message also contains the number of individuals in the group, which decides how many messages to wait for.

3. The new member waits until they receive agreements from all group members. Otherwise, the process stops.

4. The new member starts the FROST key generation process by broadcasting a *key gen commitment* message.

5. Each participant broadcasts a *key gen commitment* message after receiving the *key gen commitment* message from the new member.

6. Each participant sends a key share message to every participant after receiving *key gen commitment* messages from all participants. Each key share is unique.

7. The key share messages received are combined into a key. The key is a cryptographic key that can be used together with the keys of other participants to spend funds from the Bitcoin account.

8. Once the key generation is done, a Bitcoin transaction is created to migrate funds from the old Bitcoin account to the new one. The transaction may also contain an entrance fee from the new member if required. The transaction is described in Figure 1

Proposals are Bitcoin transactions that the group can submit. Proposals allow the group to vote on which actions to take. The process has four messages, depicted in Figure 2. This process has the following steps:

1. A member creates a proposal and broadcasts it to the other members using a *sign request* message. The message contains the proposed Bitcoin transaction.

2. The other group members respond with their votes using a *sign request response* message. The message contains a Boolean value representing the vote.

3. The proposer waits for agreements from a majority of the group before starting the signing procedure. The process is canceled if not enough members respond with agreements during the timeout duration.

4. The proposer starts the signing process by broadcasting a *preprocess* message. The message will contain the total number of participants, so all participants know how many messages to wait for.

5. After receiving the *preprocess* message, the other participants will also broadcast *preprocess* messages. This time without the number of participants.

6. Each participant creates a signature and sends a *signature share* message to the proposer after receiving *preprocess* messages from every participant.

7. The final signature is created by combining the signature shares, and the transaction is completed by adding the signature. Afterward, the transaction is submitted to the Bitcoin network. Each Bitcoin input requires a signature. Thus, the signing process may need to run in parallel depending on how many inputs there are.

Figure 4 describes the process with 2 participants.

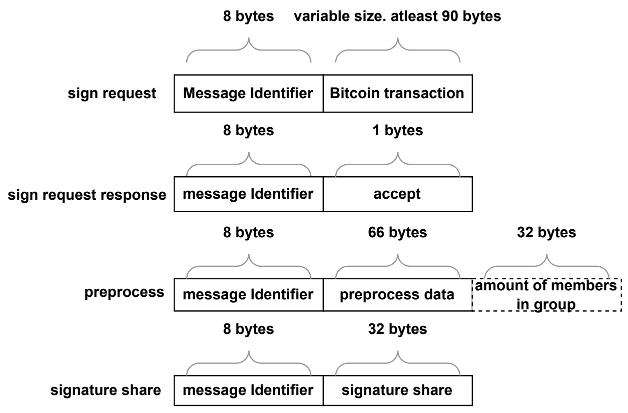

Figure 2: Breakdown of the signing protocol messages. Dashed borders represent a field that is not always required.

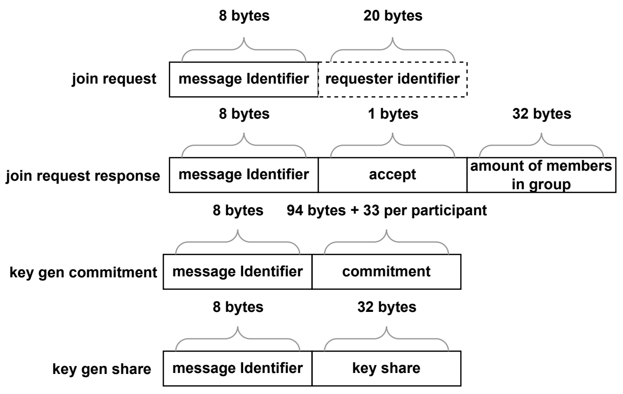

Figure 3: Breakdown of the key generation protocol messages. Dashed borders represent a field that is not always required.

In both processes, the member proposing is responsible for signaling the start of the procedure. This significantly improves performance, as otherwise, each participant would need to broadcast readiness individually.

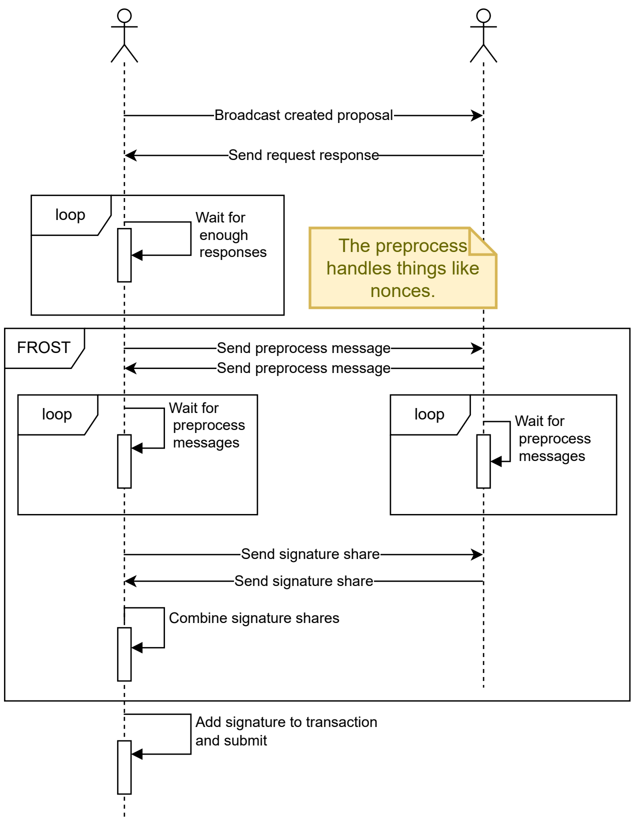
Broadcast created proposal
Send request response
loop
Wait for enough responses
The preprocess handles things like nonces.
FROST
Send preprocess message
Send preprocess message
loop
Wait for preprocess messages
loop
Wait for preprocess messages
Send signature share
Send signature share
Combine signature shares
Add signature to transaction and submit

Figure 4: Sequence diagram of the signing procedure. The diagram shows the procedure with two participants.

3.4 Identity

IPv8's identity mechanism is used to identify participants in the system. The identities are cryptographic keys, allowing participants to prove their identity by signing a message. The identity mechanism may be insufficient depending on whether the organization has anonymous members. In this case, the organization could be overtaken by a Sybil attack[19]. To prevent this, a Self-Sovereign Identity mechanism could be used[20]. Using Self-Sovereign Identity, participants could identify themselves without giving out privileged information.

Class / Package	Line coverage	Lines of code
FrostManager	93%	404
SchnorrAgent	94%	106
FrostCommunity	65%	141
FrostViewModel	0%	156
ui	0%	980

Table 1: Code coverage of the FROSTDAO application.

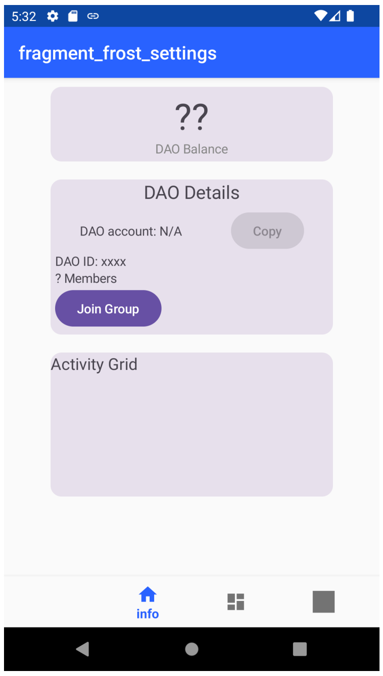

Figure 5: Android application home screen. The screen shows various details of the DAO, such as the balance and amount of members

4 Implementation

We have created an Android application that partially implements our design. Users can join a group, create a proposal, and vote on proposals. The application is open-source and publicly available on GitHub[1]. Figure 5, Figure 6, Figure 7 and Figure 8 show the various screens in the application.

4.1 Collective wealth

The Android application contains a personal wallet and the group wallet. The personal wallet is only used for testing purposes. We use the BitcoinJ open-source library [21] for Bitcoin support. BitcoinJ tracks the Bitcoins in the group wallet to allow participants to create proposals easily. BitcoinJ stores its data in an SQLite database.

To support threshold signatures, we used an audited open-source Rust library [22]. We created a wrapper around this library and then exposed the wrapper to Android via Java's native interface [23]. After the key generation procedure, each participant receives a key share. The key share is stored in a database for persistence.

Bitcoin inputs represent spendable Bitcoin. One or multiple inputs must be spent to create a transaction. The An-

[1]https://github.com/rahimklaber/trustchain-superapp/tree/frost_dao/frostdao

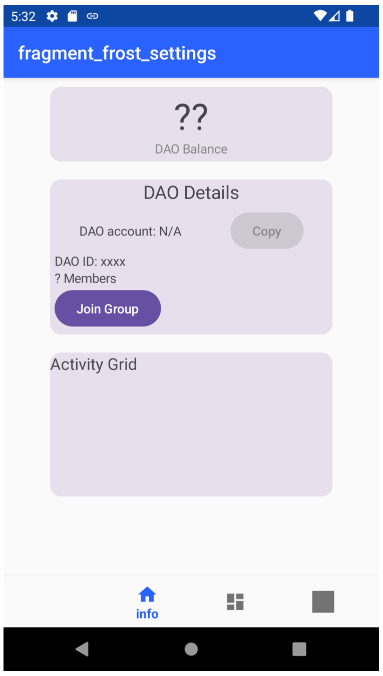

Figure 6: TODO: placeholder

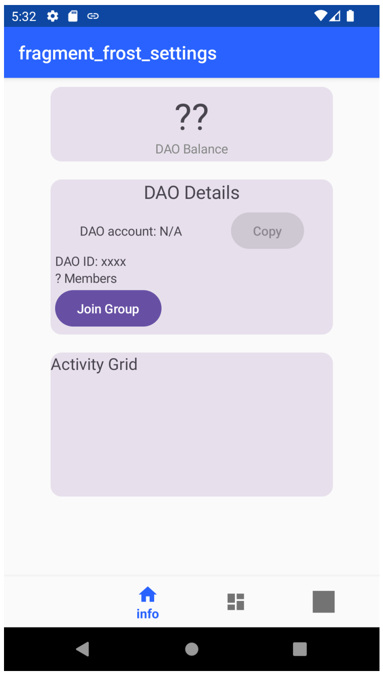

Figure 7: TODO: placeholder

droid application only supports Bitcoin transactions with one input, as Bitcoin transactions require signatures for each input used in a transaction. Additionally, only payment transactions are supported.

4.2 Joining an Organization

After clicking the join button on the screen depicted in Figure 5, The application will send a join request message and waits for responses. The joining procedure starts if enough responses are received within the timeout duration. Once the process is complete, it will be possible to view details of the group account, create proposals, and reject or accept proposals. In contrast to our design, accepting an entrance fee and migrating funds from the old account are not implemented.

4.3 Creating a Proposal

To create a proposal, A user must input the destination and the amount on the screen shown in Figure 7. Internally, a new Bitcoin transaction is created as part of the proposal. The proposal is then broadcast to the organization. Members of the organization can accept or decline the proposal on the screen shown in Figure 8. The signing procedure will start once enough participants respond with agreements. After the signing is done, the signature is added to the transaction, and the transaction is submitted to the Bitcoin network.

4.4 Quality Assurance

We used unit and integration tests to ensure the code is bug-free and the code coverage is shown in Table 1. The core of the application, which consists of FrostManager, SchnorrAgent, and FrostCommunity, has been extensively tested. However, harder-to-test code, like the UI and Bitcoin code, is not well-tested.

We used unit tests to ensure our code had no significant flaws. Specifically, we tested key generation and signing while mocking communication. Integration tests were used to test key generation and signing without mocked communication. The integration tests discovered many bugs that were caused by race conditions. We fixed many of the bugs, but some still occasionally occur. The bugs rarely occur and we are unsure if IPv8 or our code is responsible for the bugs.

We also manually tested the Android application to test the entire application, particularly the Bitcoin integration. To do this, we created our own private Bitcoin network, enabling us to create fake Bitcoins and instantly confirm transactions. We used two mobile devices for this.

4.5 Challenges

Developing any distributed system is challenging, especially fully peer-2-peer systems. During development, we encountered numerous problems and challenges. This includes challenges relating to communication, Bitcoin, and reliability.

IPv8 relies on UDP under the hood, which is not reliable.

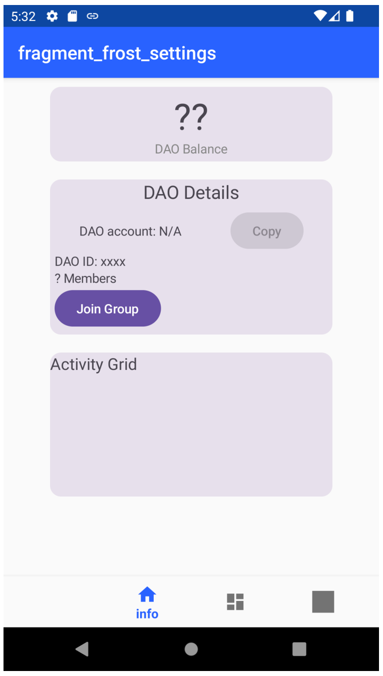

Figure 8: TODO: placeholder

To address this, we added acknowledgments and timeouts on top of IPV8. Determining the correct timeout duration and the amount of retransmission is challenging, especially for unreliable networking.

The size of the *key commitment* message (see Figure 3) during key generation scale with the number of participants. If we want the UDP packets not to be dropped, we need to limit their size to around 1400 Bytes [24], which some messages do not fit into. Therefore, we use IPv8's EVA protocol, which splits data into multiple packets. Using EVA to send messages results in high latency. Additionally, EVA transfers have a high failure rate.

The system relies on being able to send messages to all members. However, IPv8 is not meant to create fully connected peer-2-peer networks. Each peer in a community will keep track of some peers by sending periodic pings. This is not a problem in smaller networks (30 members) but becomes a problem in larger networks. This can be somewhat mitigated by changing IPv8's configuration.

The signing process requires that every participant has an up-to-date view of the Bitcoin network. However, It often happened that some participants lagged behind. This resulted in the process failing.

5 Evaluation

In this section, we evaluate the performance of our system by running multiple experiments.

5.1 Experiment Setup

We ran the experiments on a Windows 10 PC with 32GB of RAM and a Ryzen 7 3700x CPU with 8 cores and 16 threads. We modified the code responsible for communication and signing to the work in a Desktop environment with a Java virtual machine. This included compiling the native code to work on Windows. Our experiments were run in one application that created individual nodes that represent a member of an organization. Each node is an IPv8 node that runs the entire IPv8 stack. However, since all nodes are on the same PC, network latency is not a factor. We limited the number of nodes in the experiments to 50, as we ran into problems with more than 50 nodes. We modified the default IPv8 *maxPeer* configuration to allow each peer to connect directly to all other peers. Each experiment was run multiple times.

In our experiments, we are interested in the performance of the signing and key generation protocols, as these are the most expensive parts of our system. We measured the performance in two ways. First, we measured the time to do key generation and create a signature. Second, we measured the amount of data sent when running key generation and signing. This is important as we want the system to be usable on mobile devices. We further investigate by introducing artificial delays to simulate potential network delays, and we introduce random packet drops to investigate performance in a more real-life scenario.

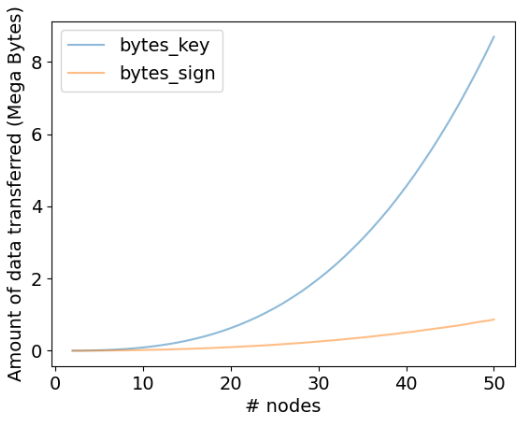

Figure 9: Amount of data in Kilobytes sent during key generation and Signing

5.2 Experiment results

Figure 9 shows the amount of data sent during key generation and signing. The amount of data sent during key generation scales exponentially-. This is expected as the size of messages sent during key generation depends on the number of participants. When running key generation with

50 nodes, each node sent and received around 150KB of data. Even this low amount of data can be problematic as everything is sent using UDP packets and may therefore be dropped without warning, resulting in even more data being sent. In contrast to key generation, the signing protocol requires significantly less data to be sent. This is expected as each signing operation requires constant data per participant.

Figure 10 shows the duration of the key generation protocol. Up to 16 nodes, the procedure has a low duration that increases a small amount when the number of nodes increases. After 16 nodes, the duration and variability increase dramatically. At this point, the size of messages sent during key generation is no longer small enough such that the UDP packets are delivered reliably. The dramatic increase in duration is due to EVA, IPV8 TFTP protocol for sending larger amounts of data. This protocol splits the data into chunks, sends each chunk via UDP, and uses acknowledgments to ensure that each chunk is delivered. EVA does not send the data immediately and instead schedules transfers in the future, which results in a large spike in duration. The large variability is due to the EVA protocol failing and needing to retransmit data and due to the protocol's scheduler. The signing protocol, shown in Figure 11, is much quicker than key generation, as the messages all fit inside UDP packets. In practice, Signing will scale much better since only a majority of the organization needs to participate. Thus, in an organization with 50 members, only 26 need to participate.

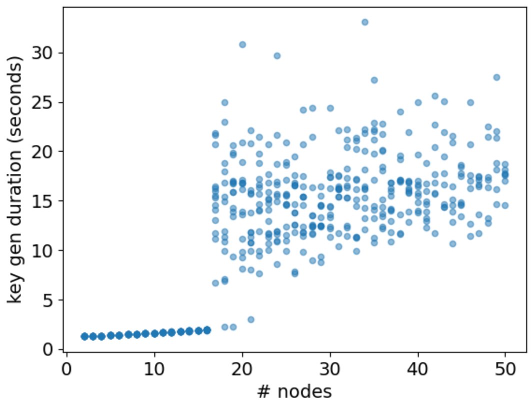

Figure 10: Duration of the key generation running on the IPV8 stack.

TODO: talk about he investigation

Figure 12 shows the duration of the key generation protocol with an artificial delay of 100 milliseconds and Figure 13 shows the average duration of the various protocols with varying delays. The artificial delay adds a constant duration to the signing and key generation protocols up to 16 nodes. After 16 nodes, the increase in duration for signing stays similar. However, the minimum duration for key generation increased by a large amount, likely due to EVA. In both cases, the range of durations is similar. But the delays have increased the duration on average.

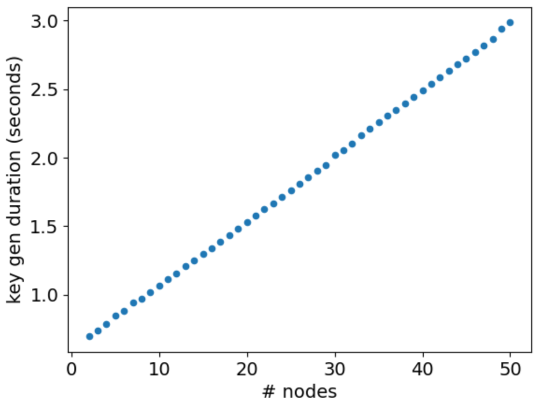

Figure 11: Duration of the signing protocol running on the IPV8 stack.

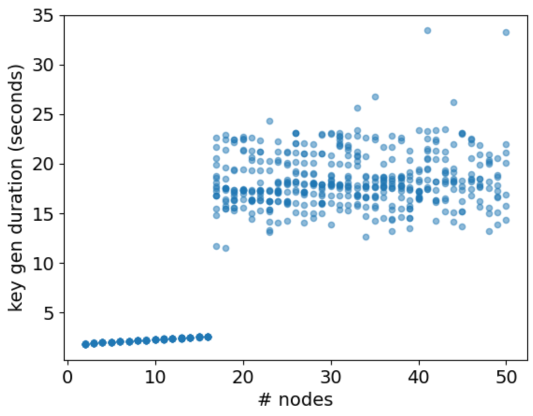

Figure 12: Duration of the key generation protocol running on top of the IPV8 stack with a delay of 100 milliseconds.

TODO: havent done this yet

Figure x shows the duration of the Key generation and signing protocols with varying levels of packet drop. We expect that the duration of both protocols will be significantly increased. This is because our message acknowledgment system has a high timeout and will therefore wait a long time for an acknowledgment. A single packet drop will result in delays in the order of multiple seconds. The high timeouts are to accommodate the usage of EVA and while it can be improved, this would lead to more complex code.

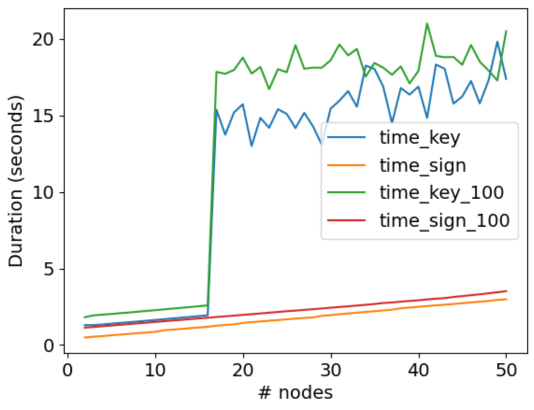

Figure 13: TODO: Add more delays? Average Duration of the key generation and signing protocols with various delays.

6 Conclusion

In this thesis, we designed a system allowing a group of people to manage their wealth collectively using Bitcoin. Our Android application uses a peer-2-peer network and does not rely on any central party, allowing the application to be used by anyone. The Android application uses threshold signatures, allowing individuals to jointly control a Bitcoin account without relying on complex smart contracts. Our expirements show that this technique is practical. Both key generation and signing take less than a minute for less than 50 participants. Our expirements also show that key generation performance can be improved by a significant amount, as the performance is being limited by IPv8.

References

[1] V. V. Acharya and M. Richardson, "Causes of the financial crisis," *Critical Review*, vol. 21, no. 2-3, pp. 195–210, 2009. DOI: 10 . 1080 / 08913810902952903. eprint: https : / / doi . org / 10.1080/08913810902952903. [Online]. Available: https://doi.org/10.1080/08913810902952903.

[2] P. K. Ozili, "Causes and consequences of the 2023 banking crisis," *Available at SSRN 4407221*, 2023.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[4] S. Lo and J. C. Wang, "Bitcoin as money?," 2014.

[5] K. Hong, "Bitcoin as an alternative investment vehicle," *Information Technology and Management*, vol. 18, pp. 265–275, 2017.

[6] D. Khan, L. T. Jung, and M. A. Hashmani, "Systematic literature review of challenges in blockchain scalability," *Applied Sciences*, vol. 11, no. 20, p. 9372, 2021.

[7] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[8] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized autonomous organizations: Concept, model, and applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, 2019.

[9] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartı, "Analysis of the bitcoin utxo set," in *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22*, Springer, 2019, pp. 78–91.

[10] S. Faraj, S. L. Jarvenpaa, and A. Majchrzak, "Knowledge collaboration in online communities," *Organization Science*, vol. 22, no. 5, pp. 1224–1239, 2011, ISSN: 10477039, 15265455. [Online]. Available: http://www.jstor.org/stable/41303115 (visited on 07/06/2023).

[11] R. Kouzes, J. Myers, and W. Wulf, "Collaboratories: Doing science on the internet," *Computer*, vol. 29, no. 8, pp. 40–46, 1996. DOI: 10.1109/2.532044.

[12] D. M. Wilkinson and B. A. Huberman, "Cooperation and quality in wikipedia," in *Proceedings of the 2007 international symposium on Wikis*, 2007, pp. 157–164.

[13] R. Fritsch, M. Müller, and R. Wattenhofer, "Analyzing voting power in decentralized governance: Who controls daos?" *arXiv preprint arXiv:2204.01176*, 2022.

[14] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "Sok: Decentralized finance (defi)," *arXiv preprint arXiv:2101.08778*, 2021.

[15] C. Komlo and I. Goldberg, "Frost: Flexible round-optimized schnorr threshold signatures," in *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27*, Springer, 2021, pp. 34–65.

[16] S. Goldfeder, R. Gennaro, H. Kalodner, *et al.*, "Securing bitcoin wallets via a new dsa/ecdsa threshold signature scheme," in *et al.* 2015.

[17] M. Skála, "Technology stack for decentralized mobile services," 2020.

[18] G. Halkes and J. Pouwelse, “Udp nat and firewall puncturing in the wild,” in *10th IFIP Networking Conference (NETWORKING)*, Springer, 2011, pp. 1–12.

[19] J. R. Douceur, “The sybil attack,” in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260, ISBN: 978-3-540-45748-0.

[20] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, “A survey on essential components of a self-sovereign identity,” *Computer Science Review*, vol. 30, pp. 80–86, 2018.

[21] P. Xiao, “Java programming for blockchain applications,” in *Practical Java Programming for IoT, AI, and Blockchain*. 2019, pp. 347–388. DOI: 10.1002/9781119560050.ch10.

[22] L. Parker, *Modular frost*. [Online]. Available: https://github.com/serai-dex/serai/tree/develop/crypto/frost (visited on 07/06/2023).

[23] S. Liang, *The Java native interface: programmer’s guide and specification*. Addison-Wesley Professional, 1999.

[24] C. Kaufman, R. Perlman, and B. Sommerfeld, “Dos protection for udp-based protocols,” in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, ser. CCS ’03, Washington D.C., USA: Association for Computing Machinery, 2003, pp. 2–7, ISBN: 1581137389. DOI: 10.1145/948109.948113. [Online]. Available: https://doi.org/10.1145/948109.948113.