FROSTDAO: Collective Ownership of wealth using FROST

Rahim Klabér

June 27, 2023

Abstract

Say that banks are bad? Present the system: it is decentralized, transparent, open-source, does not rely on any central party. Mention expirement results and say that it is practical for the values we tested?

1 Introduction

In the years leading to the 2008 financial crisis, banks engaged in excessive risk-taking for the goal of profit. They invested in risky loans using their depositors' funds and were bailed out by the government when they failed. Recently, the financial system has once again been put to the test with the failures of multiple banks. In their simplest form, banks act as a middleman between depositors and borrowers. They pool the deposits together and give out loans to other parties while taking a cut of the fee. More importantly, banks act as gateways to today's financial world. Without a bank, a person cannot easily invest, pay online or get loans.

Bitcoin emerged as an alternative to the global financial system. It gave individuals an alternative to banks and allowed anyone to securely send money to each other using the Internet. Some thought Bitcoin could be an alternative to the current financial system [1]. However, Bitcoin has largely remained a tool for speculation [2]. Real-world use is impractical due to high transaction fees, low throughput, and because it is hard to use correctly by non-technical individuals [3].

Bitcoin is lacking in several areas to become an alternative to the financial system. Some of which are:

- high throughput
- cheap payments
- privacy
- collective ownership of wealth

High throughput and cheap payments are actively being worked on with the development of the Lightning Network [4]. The Lightning Network is a protocol that lives on top of Bitcoin. It allows for cheap and fast payment by batching transactions and settling them on Bitcoin at a later point in time.

There are a number of Bitcoin mixing services that increase the privacy of Bitcoin [5]–[7]. These services mix the funds of different users by sending them to newly created wallets.

While collective ownership of wealth on Bitcoin is possible with current tools, it is impractical due to high fees and low scalability [8]. Collective ownership would allow for a group of individuals to truly be their own bank, where each individual is part owner. Money can be pooled and invested. This process is transparent and a majority of the participants need to agree for any action to be taken.

In this paper, 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 Bitcoin. Using this primitive, individuals can create shared Bitcoin accounts with hundreds of others. We achieve this without any overhead to transaction size. Our system can be used by anyone and is compatible with existing Bitcoin tools and services such as the Lightning Network and various mixing services.

2 Problem Description

The internet has revolutionized the way individuals collaborate and work towards a common goal, even across borders. However, despite this, there remains a significant challenge when it comes to the collective management of wealth. Establishing a company or making joint investments can be complicated and cumbersome, particularly when the individuals involved are from different countries. Existing financial services do not address this issue, highlighting the need for a novel solution.

The open question is whether banks and other financial institutions can be replaced by a fully decentralized and transparent system, allowing for the democratic collective management of wealth and collective investment. The challenge is constructing the system in a way such that anyone, regardless of their background, can participate and without having to rely on any central actor.

3 System Design

We aim to solve the problem by creating a peer-2-peer leaderless decentralized system. This system will allow groups of individuals to form decentralized autonomous organizations (DAOs) [9] that enable them to collectively and democratically manage their wealth. The key principle guiding our approach is decentralization, ensuring that every aspect of the system operates without reliance on any central authority or intermediary.

Our architecture includes the following four components: Decentralized communication, Blockchain, a governance mechanism, and identity.

3.1 Decentralized Communication

To support our goal of decentralization, we will use 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[10] library for communication. IPV8 allows for the construction of fully peer-2-peer networks through so-called *Communities*. These are P2P networks that contain application-specific functionality. Additionally, IPV8 employs hole-punching [11], which allow devices to communicate, even if they done have a dedicated public IP address.

3.2 Blockchain

Our architecture uses Blockchain technology as the mechanism to store and send money. Blockchain technology enables anyone to send and receive money over the Internet. Blockchain transactions are tamper-proof and can be verified by anyone. Therefore Blockchain technology supports our goal of decentralization, permissionlessness, and transparency. We use Bitcoin as our Blockchain. In our system, each group of participants jointly controls a Bitcoin account. We use the FROST threshold-signature scheme [12] to enable collective wealth management without using complex smart contracts. In theory, any Blockchain can be used with our system as long as it supports threshold signatures. Using threshold signatures together with Bitcoin allows the transactions to be much smaller and allows the system to be much more scalable compared to the traditional way that multi-signature Bitcoin transactions are created [13]. In addition, using threshold signatures means that no one can determine if an account is controlled by multiple individuals just by looking at the account and its transactions [13].

3.3 Governance

We build our governance module on top of IPV8's networking. Every action taken by the organization is democratically decided by its members. The governance module is simple and consists of a few messages for requesting membership and for creating proposals.

Requesting membership is done by broadcasting a request to all members of the organization. The prospective member then waits until they receive enough responses such that the joining procedure can be started. Becoming a member is unique in that it requires all other members to agree and participate in the process. This is because anytime the group is expanded, a new key must be generated, which requires all members. Figure 1 shows the layout of a Bitcoin transaction that is submitted when a new member joins. This transaction transfers funds from the old organization account to the newly created one. The transaction may also include an input from the new member if an entrance fee is required.

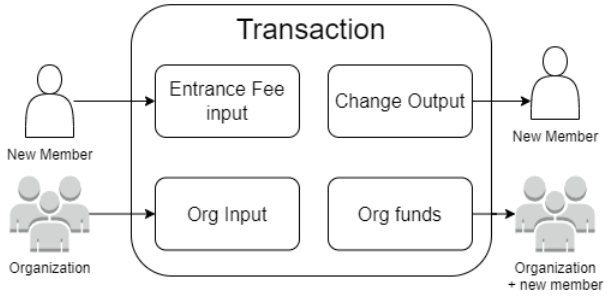

Figure 1: Bitcoin transaction for joining an organization when an entrance fee is needed.

Creating a proposal entails constructing a Bitcoin transaction that can be signed by the organization. This transaction can be as simple as making a payment or something more complex like opening a lightning channel[cite]. In contrast to accepting a new member, only a majority of members are needed to accept a proposal.

In both cases, the prospective member is responsible for signaling the start of the procedure when enough members respond. This is not decentralized. However, in this case, it does not matter, as it is the proposer whose proposal would fail otherwise. Additionally, it results in a less complex system, as otherwise, every participant would have to broadcast to every other participant that they are ready to start.

3.4 Identity

We use IPV8's identity mechanism to identify members in an organization. The identities are used to verify a member during the various procedures. Depending on if the organization is entirely anonymous or not, the identity mechanism may or may not be enough. In the latter case, the organization could be taken over with a Sybil attack[14]. To prevent this, a Self-Sovereign Identity mechanism could be used[15]. Using Self-Sovereign Identity, a participant could identify themselves without giving out privileged information.

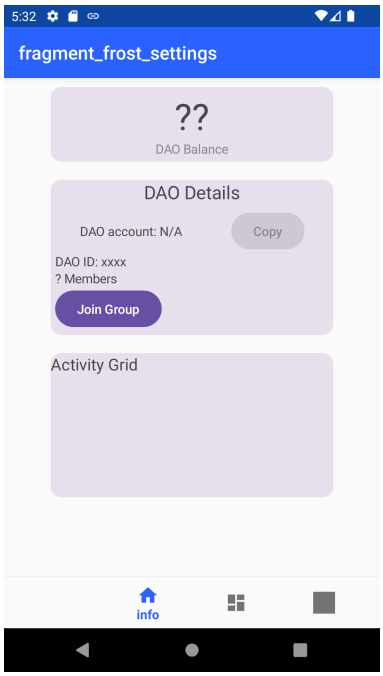

Figure 2: Android application home screen. The screen shows various details of the DAO, such as the balance and amount of members

4 Implementation

We have created an Android application that partially implements our design. Users can join a DAO, create a proposal, and vote on proposals. The code is open-source and publicly available on Github[cite]. Figure 2, Figure 3, Figure 4 and Figure 5 show the various screens in the application.

4.1 Collective wealth

The Android application contains a personal wallet and the DAO wallet. The personal wallet is only used for testing purposes. We use the BitcoinJ open-source library [16] for Bitcoin support. We use BitcoinJ to track the inputs and outputs of the DAO wallet, so that any participant can easily create a proposal. BitcoinJ stores its data in an SQLite database.

To support threshold signatures, we used an audited open-source RUST library[cite]. We created a wrapper

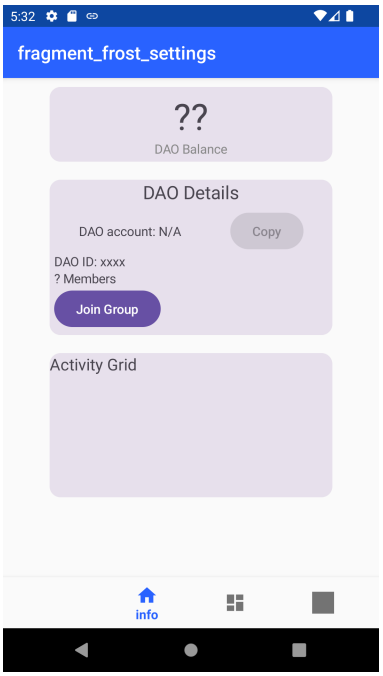

Figure 3: TODO: placeholder

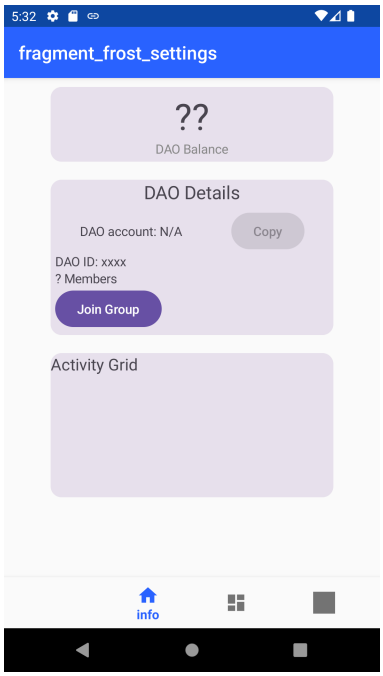

Figure 4: TODO: placeholder

around this library and then exposed the wrapper to Android via Java's native interface [17]. After the key generation procedure, each participant receives a key share, which is stored in a database for persistence. The Bitcoin address created from the threshold key is a Taproot address [18], which is the Bitcoin upgrade that enabled efficient threshold signatures on Bitcoin.

The Android application currently only supports Bitcoin transactions with one input, as Bitcoin transactions require signatures for each Bitcoin Input used in a transaction. Additionally, only simple payment transactions are supported.

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.

4.2 Joining an Organization

After clicking the join button, The application will send a join request message and wait for a response. If a response is received within the timeout duration, the joining procedure will start. Otherwise, the application state is reset. Once the process is complete, it will be possible to view details of the DAO account, create proposals, and reject or accept proposals. In contrast to our design, accepting an entrance fee is not implemented and migrating funds from the old DAO account is not implemented.

4.3 Creating a Proposal

The Android application currently only supports creating proposals to send funds from the DAO account to another account. A user must input the destination and the Bitcoin amount to create a proposal. Internally, a new Bitcoin transaction is created to represent the proposal. Once

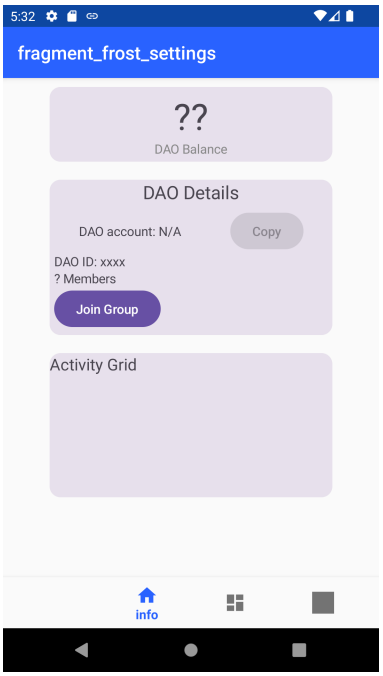

Figure 5: TODO: placeholder

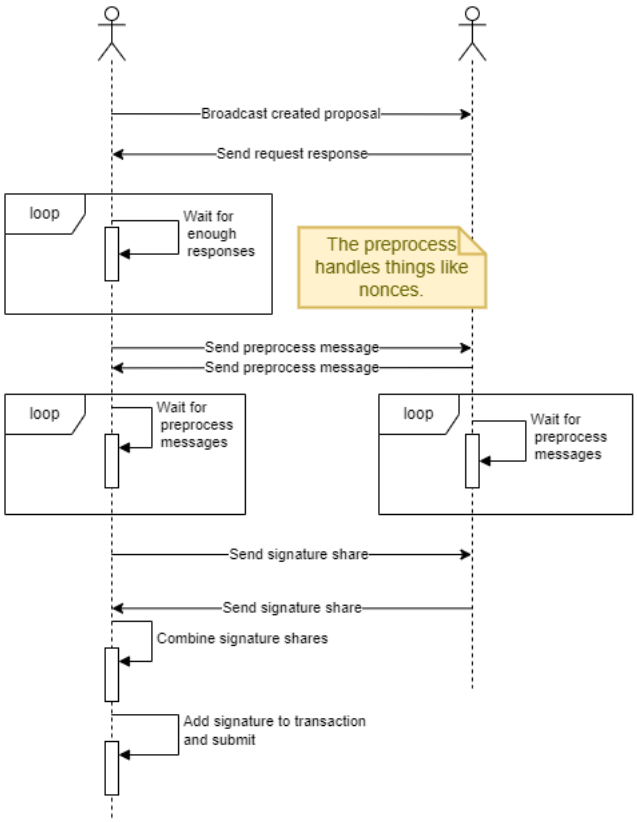

Figure 6: Sequence diagram of the signing procedure. The diagram shows the procedure with two participants.

created, the proposal is broadcast to the organization and the proposer waits for enough participants. Once enough participants respond, the signing procedure will start, after which the signature is added to the transaction, and the transaction is submitted to the Bitcoin network. Figure 6 describes this process.

4.4 Quality Assurance

We used both 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 make sure that our code has no major flaws. we used unit tests to test key generation, signing, and pre- and post-conditions. Integration tests were used to test what we tested in unit tests but without a 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. These bugs rarely occur and we are not sure if IPV8 or our code is the cause of these bugs.

We also manually tested the Android application to test the entire application, and in particular the Bitcoin integration. To do this, we created our own private Bitcoin network that enabled us to create our own fake Bitcoins and mine blocks instantly. We used two mobile devices for this.

4.5 Challenges, move to evaluation?

Developing any distributed system is challenging, especially fully peer-2-peer systems. During development, we encountered numerous problems and challenges. This in-

cludes challenges relating to communication, Bitcoin, and reliability.

As previously mentioned, we used IPV8 for communication. IPV8 relies on UDP under the hood, which is not reliable. To address this, we added acknowledgments and timeouts on top of IPV8. This works, but the challenge lies in determining the correct timeout duration and amount of retransmissions, which especially matter for unreliable networking.

The size of some messages during key generation scale with the number of participants. If we want the UDP packets to not be dropped, we need to limit their size to around 1400 Bytes [19], which some messages do not fit into. Therefore, we use IPV8's EVA[footnote] protocol, which splits up data into multiple packets, to send the messages. EVA periodically executes scheduled transfers, which increases the latency of sending messages. Additionally, EVA transfers have a high failure rate.

The DAO system relies on being able to send messages to all members of a DAO. However, IPV8 is not meant to create fully connected peer-2-peer networks. Each peer in a community will keep track of a number of peers by sending periodic pings. This is not a problem in smaller networks (30 members), but it 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. Due to this, the signing process sometimes failed.

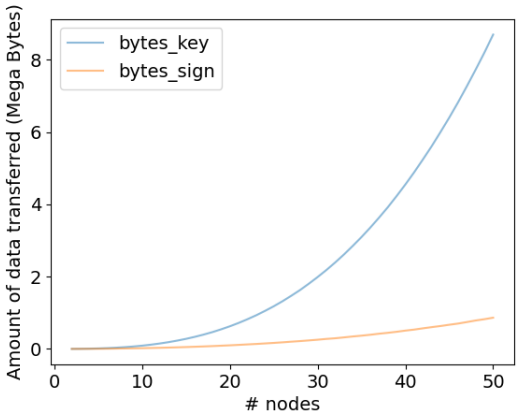

Figure 7: Amount of data in Kilobytes sent during key generation and Signing

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 that has 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 was responsible for creating the individual nodes, that each represent a participant in the DAO. Each node is an IPV8 node that runs the entire IPV8 stack. However, since all of the 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 it takes to do key generation and to create a signature. Note that in the case of signing, we are doing the 2-round procedure and not the optimized 1-round version. Second, we measured the amount of data that is 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 drop to investigate performance in a more real-life scenario.

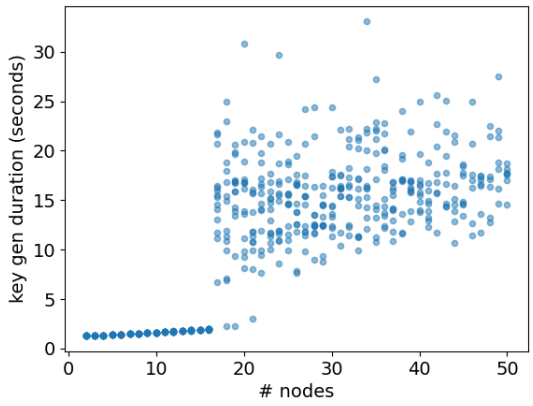

Figure 8: Duration of the key generation running on top of the IPV8 stack.

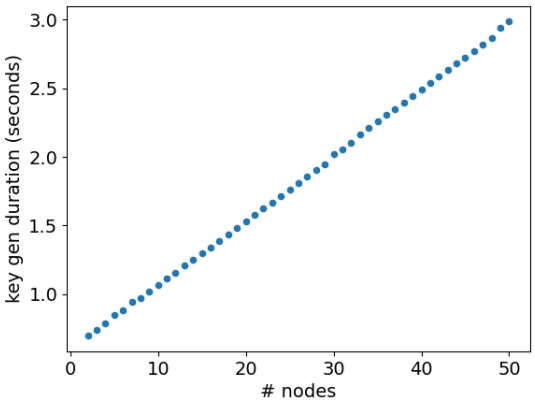

Figure 9: Duration of the signing protocol running on top of the IPV8 stack.

5.2 Experiment results

Figure 7 shows the amount of data sent during key generation and signing. We notice that 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. While the graph paints a bad picture, keep in mind that this is the total data sent and not only the data sent by one node. This still means 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 a constant amount of data per participant.

Figure 8 shows the duration of the key generation protocol. Up to 18 nodes, the procedure has a low duration that increases a small amount when the number of nodes is increased. After 18 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. Attempting to use UDP packets at this point will result in them getting dropped. 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 9, 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 shows the duration of the key generation protocol with an artificial delay of 100 milliseconds and Figure 11 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 18 nodes. After 18 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.

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 acknowl-

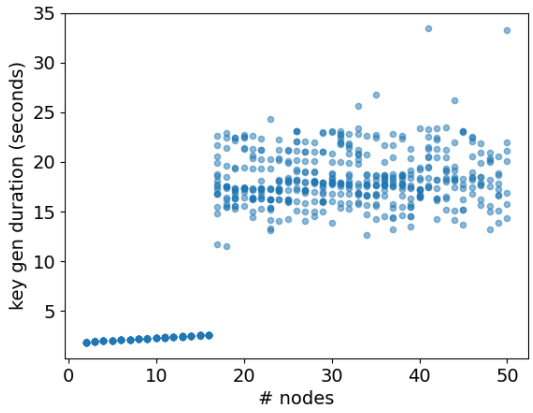

Figure 10: Duration of the key generation protocol running on top of the IPV8 stack with a delay of 100 milliseconds.

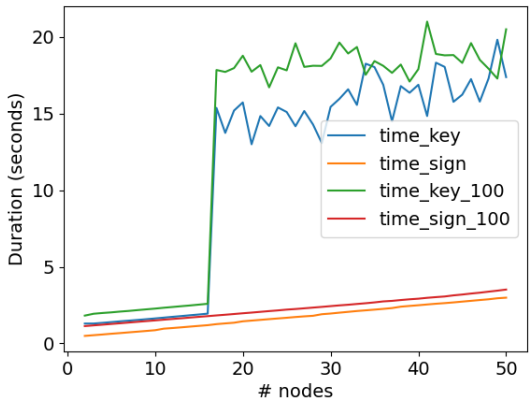

Figure 11: TODO: Add more delays? Average Duration of the key generation and signing protocols with various delays.

edgment 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.

6 Discussion

Our results show that FROSTDAO is practical for the organization sizes we tested. While key generation can take up to 30 seconds, this is not a large duration, when we consider that key generation is only done when a new member joins. On the other hand, signing is extremely quick.

Different latencies do not change the duration of the protocols by much. This is likely because many of the messages are sent concurrently, and thus a larger latency will not affect much.

The large amounts of data sent during duration key generation is concerning, especially with how it seems to scale. This, in addition to IPV8's overhead, is what is holding us back.

7 Conclusion

References

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

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

[3] A. W. Baur, J. Bühler, M. Bick, and C. S. Bonorden, "Cryptocurrencies as a disruption? empirical findings on user adoption and future potential of bitcoin and co," in *Open and Big Data Management and Innovation*, M. Janssen, M. Mäntymäki, J. Hidders, *et al.*, Eds., Cham: Springer International Publishing, 2015, pp. 63–80, ISBN: 978-3-319-25013-7.

[4] J. Poon and T. Dryja, *The bitcoin lightning network: Scalable off-chain instant payments*, 2016.

[5] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "P2p mixing and unlinkable bitcoin transactions," *Cryptology ePrint Archive*, 2016.

[6] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *Financial Cryptography and Data Security: FC 2015 International Workshops, BITCOIN, WAHC, and Wearable, San*

Juan, Puerto Rico, January 30, 2015, Revised Selected Papers, Springer, 2015, pp. 112–126.

[7] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, Springer, 2014, pp. 486–504.

[8] 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.

[9] 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.

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

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

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] P. Wuille, J. Nick, and A. Towns, "Taproot: Segwit version 1 spending rules," *Bitcoin Improvement Proposal*, vol. 341, 2020.

[19] 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.