medixain: Robust Blockchain Optimization Enabling Individual Medical Wallet Architecture

Simon Schwerin, medixain GmbH
Amin El-Kutbi, medixain GmbH
Leif-Nissen Lundbæk, XAIN AG
Felix Hahmann, XAIN AG
Laurence Kirk, XAIN AG
Daniel Janes Beutel, University of Oxford
Michael Huth, Imperial College London

November 22, 2017

1 Introduction and Problem State

1.1 The Problem of Inherent Fragmentation

In todays rapidly aging societies further medical advancements more than ever rely on the automated analysis of vast amounts of data to receive more significant results of, e.g., medication effects. Particularly in medical research, it is evident how important such procedures are, keywords are, i.e., next generation sequencing or imaging techniques, yet, also medical practices strongly require such tooling. A major challenge in the health sector is demographic change, with an aging society and the resulting increased morbidity of the individual. At the same time, medicine controls from a reactive to a predictive and preventive medicine which, through early interventions, attempts to prevent illness or at least to treat it at an early stage as well as also highly individualized. This paradigm shift from a global medicine to an individual, stratified medicine should lead to the individual being able to enjoy good health. This is the only way to combat rising costs in the health system.

To achieve this, it is necessary to collect and combine large amounts of heterogeneous personal data. These include, for example, clinical, epidemiological, imaging, molecular genetic, but also economic data. By analyzing such data using "data mining" techniques, new knowledge about disease development, prevention and individualized therapy can be generated. At the same time, economic data can also be used to develop models which, in addition to an optimization of medical care, can also show cost savings. Since the hospital information systems mostly include economic and clinical data, the prerequisites for the development of models are presented, which show an optimized diagnosis and therapy and at the same time improve the economic conditions without sacrificing medical quality.

Moreover, not only medical institutions but also patients themselves suffer from a lack of sophisticated access possibilities to their medical records. Although we have the means of highly digitalized processes, patients would need to collect an entire suitcase

of various printed medical documents from different institutions as well as medical doctors. Thus, this prevents, e.g., an easy migration to new medical doctors in cases of moving to a different city or even of certain treatments where highly specialized doctors are needed.

However, current centralized solutions to electronic medical healthcare are unsophisticated, meaning they are unable to handle the large scale demand on their infrastructure faced daily. They do not incorporate into their model an easy interaction layer from which patients using such services can access their records. Instead, medical records are trapped in a closed isolated silos from the users and commonly even other medical professionals. As such, each medical institution, particularly regarding hospitals, has its own individual legacy system, resulting in a high degree of fragmentation with costly means of integratability. Data sets of individual hospitals, however, are often too small and, thus, too biased to gain significant results of treatment or medication effects.

In order to overcome this situation, hospitals increasingly try to integrate their data sets on huge external databases or data providers, such as the developed by the SAP Innovation Centre Potsdam and the Hasso-Plattner-Institut [23]. Yet, this data integration process faces common data warehousing issues, such as highly costly data cleansing and data preparation of highly different data systems with varying levels of aggregation as well as representation. Further, such solutions face an enormous overhead to overcome the intrinsic single-point-of-failure problems of such data storage systems. Such medical databases of course hold the respective data redundantly, however, this only pushes the single-point-of-failure towards the database management and integration process with partly human-based mechanisms. The storage of vast amounts of highly sensitive data, furthermore, also results in a high attractiveness for various attacks and errors, including impersonation attacks as well as data breaches, affecting millions of patient records.

In addition to this, several models exist for health care systems managed by governmental organizations. These can be seen in European continents, such as the United Kingdom with National Health Service (NHS) or Germany with Krankenkassen. These systems benefit from already having a central body that funds and oversees standards and practices. Therefore, they should expect to be receiving a complete feedback of the data that is being fed into the system they are funding. This data can then be used to optimize the systems and create better treatments. Unfortunately, these systems have a large problem of communicating this data. They were not build with the interaction of new technologies and web infrastructure in mind, they have merely been pieced together.

1.2 Information Asymmetry and Ownership

A further problem of medical data systems arises from the question of data ownership. In any modern healthcare system patients are legally provided with many rights over their medical records and often high privacy rights are in place, however, patients are

neither the ones who have an easy access to their (highly fragmented) data nor even store such records. As such, the entire data storage is pursued by medical doctors, hospitals, health insurers, pharmaceutical researchers or even intermediary third party institutions that collect and sell this data. Thus, patients have neither means of controlling where their data is stored nor of knowing how these records are utilized. Additionally, in many cases they receive no monetary benefits of such data trades at all, although they are the actual owners of the data, leading to a high extent of asymmetric information.

Furthermore, while patients face such a lack of control, medical and pharmaceutical institutions only have a limited possibility to actually verify the history and veracity of the received records, which can vary in a broad bandwidth, particularly when engaging with third party data providers, resulting into less significance of pursued studies. Such variation may result from fraud as well as common errors in the data integration process. Nonetheless, many organizations, especially in the pharmaceutical sector, are facing the need to establish study groups of newly patented drugs as fast as possible, in order to introduce it more quickly into the market. As such, their logical aim is to expand the time of product sales under the protection of the patent, as prices drop immensely afterwards. However, the identification of testing groups and the entire setup of several study periods as well as the receiving of further medical data records to underpin their position takes several years in most cases. Therefore, a direct way of corporate user interaction on a possibly anonymous but verified instance, including the financial rewarding process would be most beneficial for such institutions.

A further issue around medical records and trust arises from the interaction between patients or generally users, which is mostly pursued on various Internet forums to discuss medications, treatments and anything around the basis of medical records. Hereby, the actions of users is highly influenced by an unverified and untrusted communication. In most cases there is no opportunity to verify that other parties even suffer from a certain condition, which, however, can be seen as a minimum level of experience in an unprofessional sphere. Furthermore, this makes it hard for patients to actually find answers and other people with a certain condition to share their experience. Thus, forum searches are often a very unsatisfying and complicated procedure to find such counterparts, even if they were to be trusted.

1.3 Outline

Now, to address the named problems above this paper introduces an application framework that makes use of the current advances in distributed ledger technologies, most importantly Blockchain technology. As such, we provide an introduction into systems in general and a reasoning of technical choices to provide a legally sound consortial infrastructure solution. This system approach is particularly focussed on an individually distributed storage of medical records on the patient side. As such, Blockchain technology in general offers the possibility to achieve a consensus between distributed parties on protocol level and, particularly considering the development of the Ethereum

corporate infrastructure, we have the possibility to implement distributed applications as well as a layered storages with different levels of aggregation on top of this structure.

Introducing the architecture of the layered medical wallets as well as its levels of confidentiality, we also provide a reasoning and architectural introduction of the user interaction model and its implications. Further, we demonstrate how (pharmaceutical) corporations and medical institutions benefit from directly interacting with patients on protocol level, based on the different aggregation level of the wallet system. This also involves the actual business model of such an approach.

Having defined the distributed application itself as well as the different interaction models, we also provide our technical and mathematical developments of the entire system architecture and the introduction robust optimization as well as machine learning engines as part of the network itself. We will demonstrate that the latter optimizes the security and stability of the network in many dimensions and actually enables the corporate user interaction model of the Blockchain-based application. As such, we introduce *medixain* as a highly optimized and layered storage architecture running on top of the Blockchain Intelligence Framework as a machine learning or more specifically reinforcement learning optimized Ethereum enterprise network. Having introduced the mathematical and architectural specifications of the framework, we also test and validate the application and the network, as a proof of concept.

2 Individually Layered Medical Wallets

2.1 Introducing Blockchain Technology

To understand Blockchains we first of all consider a completely distributed network of nodes, representing human or machine based access points with storage capacity. These nodes want to establish a shared and trustless data ledger without a trusted party as a single-point-of-failure. As such Blockchain Technology serves as a perfect solution to the Byzantine Generals' Problem in distributed systems, where the communication of nodes can be impersonated and, thus, not trusted. In order to resolve this problem and to achieve consistent states in the entire network without a central administrator, Blockchain systems apply a combination of techniques in a shared protocol.

The protocol, also graphically demonstrated in Figure 1 starts with the transmission of new data from a person or machine to another node. This data is then spread in the network and collected by so-called miners as special nodes. The following procedure is called mining using a Proof of Work (PoW) algorithm: Miners bundle all transactions of a certain timeframe, order them in a tree structure and try to compute a hash value of this structure, including the hash value of the last block in the history and a timestamp. Further, this hash needs to satisfy the condition of a nonce value as a certain number of zeros for the first values of the hash to increase the hardness of finding a suitable hash by brute force guessing.

All these information are then represented as a block. As such, we achieve a com-

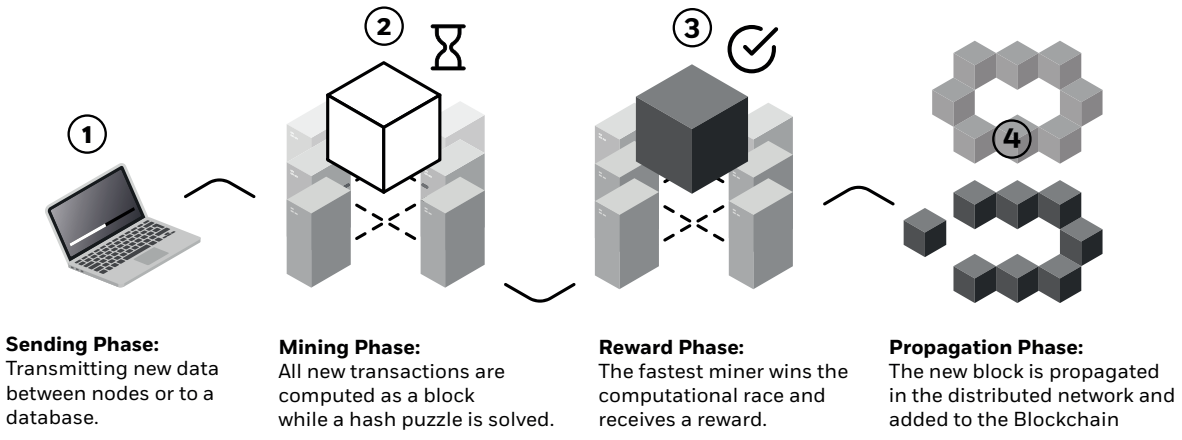

Figure 1: Blockchain Mining and Propagation Functionality

putational race, where the miner with the first block that satisfies the conditions wins the competition and receives a reward. In open systems, such as Ethereum, this reward is often monetary, to compensate for energy cost and to insert incentives for miners to invest computing power. Through this race we gain a pure randomization, as the attacker does not know, which miner to attack in advance.

Furthermore, all miners propagate their new computed block in the entire network. Nodes individually pick the fastest block and add it to the Blockchain. This way every node views eventually the same system state, as the longest chain always wins. The Blockchain achieves immutability of data, as attackers would need to change data and simultaneously beat all participating miners in the race. Thus, the adversary requires more than 51% of the entire computational power to change data in the long term, which comes at significant cost in open systems.

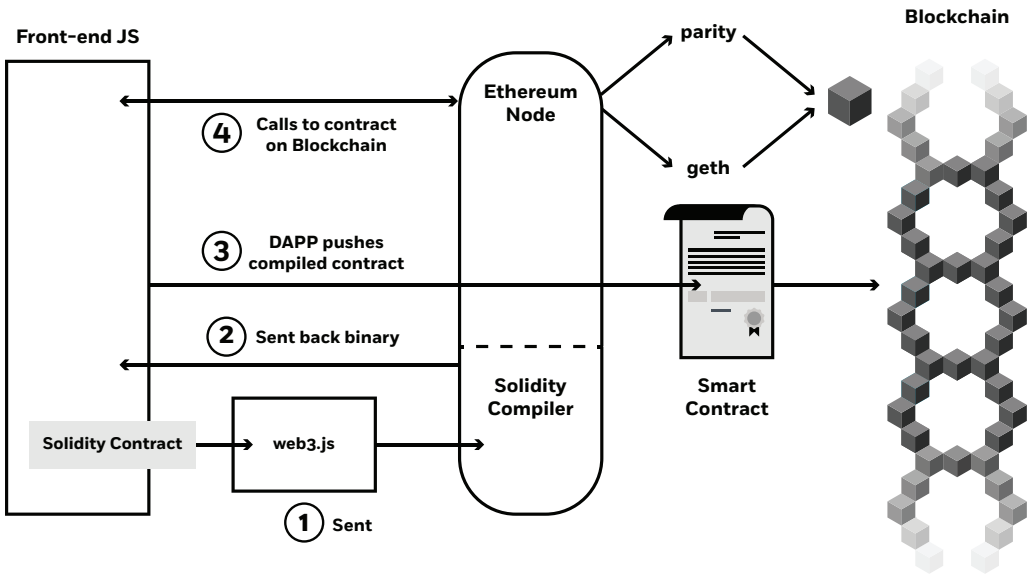

Figure 2: Ethereum Architecture

Second generation Blockchain systems, such as Ethereum, shown in Figure 2, offer the possibility to write and run distributed apps, so-called Smart Contracts, in a consensus on all nodes. They offer computational logic, executed in an automatic and verified logic between different parties without a central instance. Therefore, Blockchain systems are perceived as a revolutionary technology, similar as the Internet. They offer complete new business models, such as non-administrative payments, like Bitcoin, as well as immense improvements in secure automations of various distributions of existing systems, applied, e.g., in areas of auditing, supply chains or insurances.

However, the strength of open Blockchain systems cannot be easily transferred to enterprise or consortial systems, as they do not allow for an access control or even a Know Your Customer (KYC) policy. The latter, however, is a strong part of our approach to use Blockchain technology for a medical layered architecture. Therefore, our system involves a controlling layer built into the Blockchain to permission its access functionality. Hereby, the core of a permissioned Blockchain system is the same as unpermissioned Blockchains. With the exception for one additional layer or engine that operates the access control scheme in a distributed consensus over the entire Blockchain.

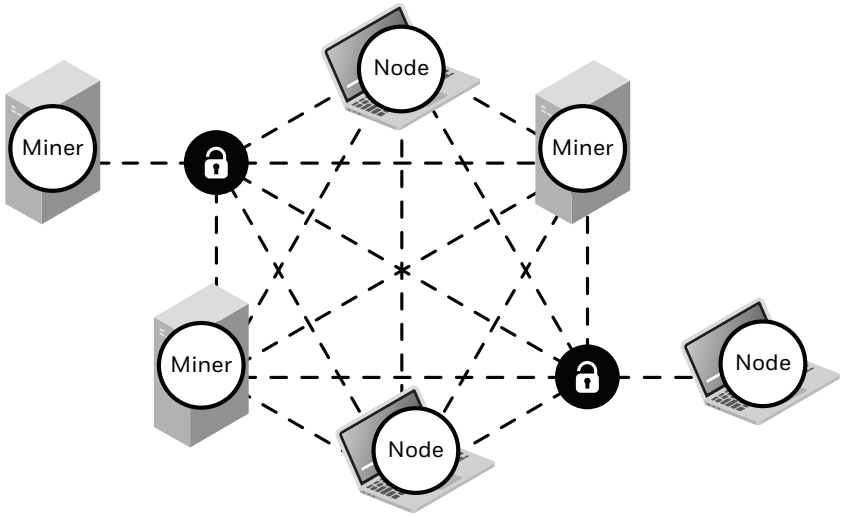

Figure 3: Permissioned Blockchain System: Nodes and Miner

The access control functionality itself, however, can differ in various ways, also shown in Figure 3: First of all, it can mean whether users in the system have the means to actually participate in the Blockchain by being a full node or having the ability to create and operate new smart contracts as a programmatic logic. The permissioning of node participation is thereby relatively easy to achieve. Secondly, permissioning the Blockchain network can also mean to influence the mining architecture itself, including the possibility of nodes to participate in the mining game. This control functionality is much harder to achieve, as it requires encapsulated functionalities of the running Blockchain clients, in our case Parity of Ethereum. This requirements mainly results from the possible ability of nodes to impersonate miner addresses, such that we need to change the actual structure of the block headers to include additional encryption and validation procedures of further parameters, which we define more closely in the optimization sections of this paper.

2.2 Introducing the Layered Wallet Architecture

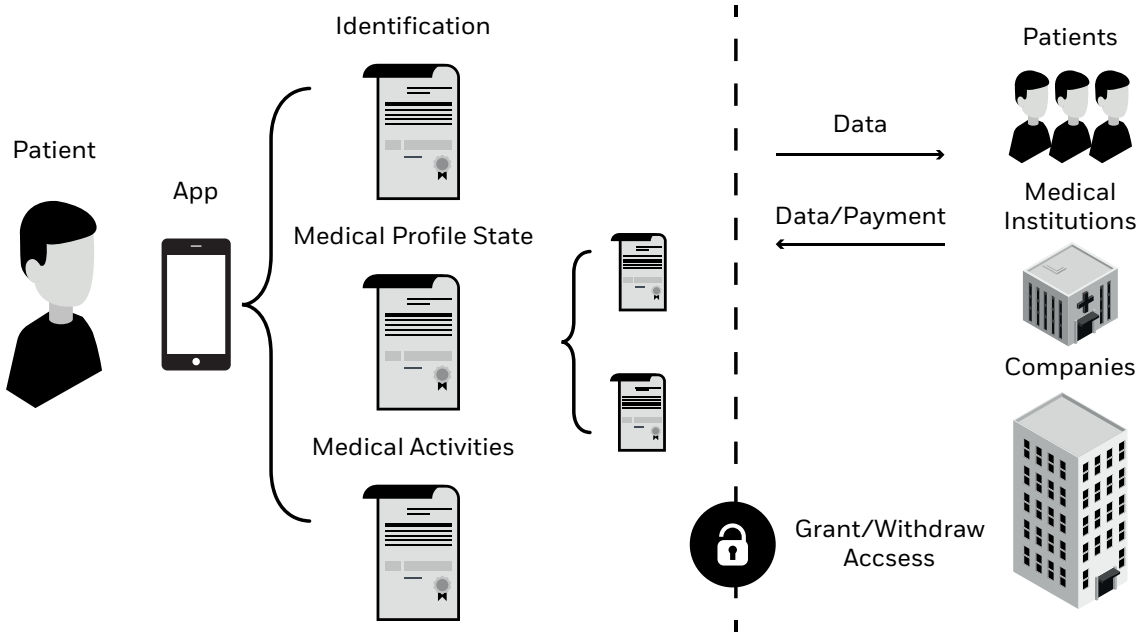

Figure 4: Permissioned Blockchain System: Nodes and Miner

To create a solution that upholds the high sensitivity and demand on medical information a multi layered approach is proposed, as shown in Firgure 4. Three core layers are identified as being separate wallets. However, this does not mean that we could not integrate or establish more layers, if needed. Each wallet has a specific level of aggregation for different purposes, while they together function as an integrated system with seamless user interaction, such that the user handles, e.g., only one mobile application as the access to the system. As such, we can introduce a mapping towards specific corporate interactions, such as for pharmaceutical research institutions, if the patient accepts the respective conditions. Regarding the core layers, we introduce the following as part of each application: An identification layer, a medical profile state layer and medical activity layer. These layers (or individual wallets from an architectural view) could furthermore also have additional sublayers that needed to be granted in privileges, to ensure the anonymity and, e.g., to ease the the medical professionals diagnostic job.

Finally, these wallets run as bundled distributed applications on a combinatorial system. The first part is consisting of an advanced Blockchain network (Blockchain Intelligence Framework), in order to enable the distributed data consensus in the network as well as for establishing the possibility of a functional access control mechanism, e.g., to grant access to certain medical doctors. This also includes the logging and analysis of the entire access history. This network works as an advancement of the Ethereum Enterprise System and integrates the possibility to permission parts of the system, e.g.

to integrate the needed KYC strategy when interacting with legal medical institutions. Further, it integrates machine learning engines to optimize the network parameters on run time for different purposes: One example is intrusion detection and the most important engine applies Deep Reinforcement Learning to stabilize the network and to increase its security based on current circumstances. The latter will be specified more deeply and mathematically in later sections. The second part is the actual storage of the medical records. This is handled by the Interplanatory File Storage (IPFS) system, in order to achieve the needed scalability of large amounts of data, which we need in this case. IPFS is hereby interchangeable with other systems, however, offers a better integration than, e.g. Cloudera, as well as a greater stability than newer approaches, such as the Interplanatory Database (IPDB).

2.2.1 Identification Layer

Regarding the wallet layer architecture, we start the entire process with the Identification Layer. This structure, on the process of generating their medical wallet, affiliates standard identification information about the user or patient to the facilities. These components would consist of personal information, such as fullname, address, national insurance number, birth certificate details, family relations, bank details etc.

These details are key to ensuring a validation of the individual on wallet generation and also play a small role regarding medical health. That individual's personal health care professional would most likely be given full access to this data. Using this they can then have contacts in case of emergency situations and have the full picture considering they have the most responsibility to that patient. The important factor here is that when the patient would usually disclose this information to a specific health care practice they would be then allowing every professional direct access. Instead individual professionals they want to view the data to be given would have that control, preventing data leakage through new team members and attacks on that isolated medical practice.

However, this layer is only activated, if the medical professional is registered in a certain access control level that he achieves by being authorized by the underlying medical institution (hospital, health insurance, etc.) and being trusted by the patient. This trust level is supported by the fact that patients can easy grant such access but also view the access log as well as withdrawing the access possibility of a specific medical professional. On the other side, medical professionals can trust the existence of a medical wallet as the data log of certain patient by verifying the hash-based wallet ID on the Blockchain, as this requires a KYC proof by, e.g., an underlying health insurance.

2.2.2 Medical Profile State

The second required layer is the Medical Profile State. The medical diagnostics of a patient can be summarised into key profile aspects. A patient would have data on bloodtype, sex, allergies, diseases, conditions etc. These are independent of the identification of the patient and also the medical activity of the patient. These pieces

of data can be shared in a controlled manner and would not expose the true identity of the patient. However, of course with modern data analysis abilities the exposure of this data still means that there is a chance of identification. Thus, this layer, consisting of the actual raw data, shall only be viewed by the patient and the medical professionals. The latter also have the means of amending new data to the storage, while patients only have partly such rights in specific predefined sections, to insure the data validity. Further, each update is signed and does not overwrite the previous state.

As such, this structure opens the door for independent doctor diagnostic and performing analysis on health combinations with certain conditions and diseases and share of discussion on the diagnostic experience without exposing identity. All such activities are completely logged and can be analyzed by the patient. Thus, hospitals, e.g, can perform much wider as well as specific data analysis of medications and treatments than it is currently possible, levereging advances in medical science and treatment effectiveness, while giving patients more rights. Moreover, the users of these wallets can define themeselves which information they want to show, such that they can easily move to another city or just change their medical professionals, e.g., in circumstances of lost trust and therefore withdraw the previously given access.

This approach could then have deeper layers to expose on a consensus basis, such as requesting dates of diagnostic or preparations received as well as variations of medication influences on the raw data, which could be, e.g., interesting for pharmaceutical corporations, while it exposes much less data that could lead to unique identifications.

2.2.3 Medical Activities

A final core layer is related more to the practice patients are receiving and by whom. These contain details of recent appointments, requests on data by whom and edits on data by whom. This layer could be used to assess the quality of practice being given to the patient and the levels of safeguarding standard being upheld to this individual. This can be done without exposing an individual's identity and the specific healthcare they have been given. Furthermore, it can be used to optimize the health routines for that individual, notify about new checkups required or see trends of doctor working with patient data. As such, it supports the informative approach of the Medical Profile State layer to optimize the quality of the entire system and grant patients more rights.

3 Interaction Models

Now, having introduced the basic structure of the medical wallet layers, we discuss how we can apply this system for different ways of interactions concerning the patients themselves as well as the binding towards the respective corporations that are either part of the system or may profit from using it. The latter will also describe the business model behind the system itself and how both patients and corporations can profit from it besides having higher privacy and a better integratability.

3.1 User Interaction Model

In the first part of this section, we discuss the question of how patients can actually use their wallet and the entire infrastructure for their own good.

3.1.1 Secure Evidence of Treatment

Once access and control is given into the patient's hand, significant increase in the service they provide will become available. Individuals are able to access their data quickly and securely from anywhere, using only a mobile app, which seamlessly bundles the respective wallets. This also opens up transparency layer of the interaction with their data. Giving patients the insight into if they are getting the correct interaction they require. They can monitor the changes and become a more informed party.

They will also have now direct access to legitimate evidence of these diagnostics. This could be used as evidence for insurance providers to know exactly what treatments they should be paying for. This additionally gives the user the evidence for claims, if a malpractice situation arises. This should speed up the process of these claims as the evidence is already created. Further, it also relieves the burden on infrastructure cost that would of been needed to investigate these medical claims.

Moreover, as previously described, patients can easily grant and withdraw access towards their medical records, allowing, in combination with the complete access log, for a much better control of medical activities and putting pressure towards professionals to behave well. Last but not least, patients also have the means to get rewarded for certain behaviors. This can include to proof regular medical checks as well as the possibility to actually trade own medical data, e.g., to studies of medical research institutes, which then have the means to directly transfer monetary rewards.

In addition to having access to the medical history, the patient can get the security and immutability alongside this. Patients would then be able to build up a medical history that should not be lost and if lost have a secure blame network in place. This system can become important upon older age where individuals might see reemerging diseases such as asthma – the specific details of the diagnostic at that time can then be reviewed.

3.1.2 Open Trusted Medical Forums

Having a layered confidentiality further also allows for trusted discussion among peers in an anonymous setting. Conditions, diseases and allergies can be discussed for tips and services that work best. Considering the popularity of sites such as webMD there is an overwhelming need for users to check first about their symptoms. However, none of these services are verified. There exists no amount of trust in any of the advice given in those forums. By building this communication layer on top of a medical wallet, trust can obtained and certainty is given that they are discussing these issues with a person who has experienced the same procedure they have. This does not replace the need for a medical examination, this just adds a layer of discussion. As such, it should help

ease the patient into receiving the correct service without depleting too much of the professionals time.

This would also then open up a new layer of data to analysis when patients are receiving treatment. These forums would have open truthful discussions that can be processed to rate the understanding and explanations of the healthcare they are receiving, without endangering the identity of any of the individuals. The latter is achieved by a adding a sublayer (technically an additional wallet in the bundle), which only contains a very high level aggregation of the medical condition, in order to maintain a high level of privacy. Patients can then decide whether or not they want to include certain verified information or whether or not to participate in this structure at all. In case they have the aim to actively discuss their conditions, they have the means to identify other system members based on aggregated searches but only see the respective hash value as the identifier of that user.

3.1.3 Medical Professionals

The further benefits to user side exists strongly in the medical professionals. They will then have a rich resource to examine practices in the past and learn from this. Whereas before they only have selected clinical trials to review.

In addition, the medical professional community collaboration when diagnosing and treating a patient can be increased. With this platform to involve new people into the discussion a address would just need to be shared and correct privileges added. Clear logs can then be seen to honor those professionals that have seen to be helping with the medical examination. Improving the incentives, medical professionals may receive rewards for following good practice. This system can additional allow for analysis of professionals that may be spread too thin over patients and work can then be done to decrease their workload.

3.1.4 New Confidence Data

First of all, as a direct by-product of having individuals accessing and controlling their data, a new confidence can be added to this data based on that agent's inspection. Having patients constantly monitoring the state of their medical record, any inconsistencies in input can be brought up from the patient. These consistencies could be professionals not filling into reports correctly through misunderstanding or misdiagnosis. This can further be argued that no one would care more about the accuracy and precision of their data than patients themselves. The professional has a duty and job to provide the best service, however it is often the case where such professional has many patients to attend to and can not physically possible ensure complete accuracy in their input over long period of times. Thus, this data integrity increase, reduces the costs of wrong data.

3.2 Corporate Interaction Model

The following subsection introduces the benefits of businesses as well as medical institutions of interacting in this architecture. These benefits may include monetary but also additional ones, such as the ability of wide-scale data analysis.

3.2.1 Institutional Verification

The first corporate or rather institutional interaction point clearly lies in the actual architectural setup. The wallet structure itself gains significantly from having a KYC strategy filter. This means that a certain legal entity verifies the truthful existence of a patient and sets up the wallet. This approach permissions the system in that way that every person may only have a single wallet, while it also offers the ability to reset, e.g., a lost password, such that no data is lost.

This approach has the advantage that we can not only ensure the recovery of data but also have a much stronger verification in the system itself with a much finer grained access control functionality. Thus, medical professionals, e.g., have more rights on accessing raw data than a pharmaceutical corporation, even if the patient accidentally granted access to everything. Furthermore, the legal entity can also ensure a certain data quality and integrity.

Finally, these institution(s) have the means to operate the system in groups, which we mathematically demonstrate in a later part of this work. This group-based integration allows to apply the architecture on top of existing infrastructures, such as heath insurers, which in some countries, like Germany, are highly fragmented themselves and have, thus, a lack of integratability of data. As such, if considering health insurers as the KYC owners, they have the opportunity to profit from a much better data integration and analysis of patient behavior. This allows them, e.g., to reward good behavior, e.g., with a reduction of fees or the participating in various course programs.

3.2.2 Analysis on Complete and Large Datasets

Upon treatment of the fragmentation issue that surround the segregated regions of current healthcare systems, it becomes difficult to analyze on a complete picture. Having an incomplete picture of data leads to an incomplete analysis. With using the decentralized govern protocol the data has been stitched together and location of this data can be found. Analysis can then be performed on a complete dataset. Further, this system also has the benefit that medical research institutions, including hospitals, can directly contact a much larger quantity of patients to ask for data access for giving a specific reward. Thus, they have the ability to overcome the currently often appearing lack of significance in the results of study as well as the restriction of smaller hospitals to pursue such studies, due to having only small numbers of patients with certain conditions. Naturally such studies may come with certain cost for the institution, in order to reward the patient, although there is also the possibility of non-monetary rewards, including also the integration of health insurers, overall it can reduce the cost, as it

decreases the need for large data warehouse solutions significantly. The latter often require vast amounts of data cleansing and integration cost, while they are also affected by security issues and the general single-point-of-failure aspects. Overall, this analysis can be used to give individual patients better treatment, cross reference the status of similar patients and review the performance of medical professionals in the field.

3.2.3 Pharmaceutical Studies

A further corporate interaction level consists of pharmaceutical studies. These corporations usually have the requirement to reduce the time of officially testing a new drug after having patented the medication. This is a logical process, since only during the existence of the patent protection of 20 years they can sell the drug for prices that cover the huge development expenses they had upfront as well as to make a certain a profit to invest in new developments. However, the selling period may only start after an official grant, which itself requires the proof of extensive tests. These studies, yet, often take roughly seven years and, thus, almost halve the period these organizations may sell the drug. This in turn diminishes the possible profits and, hence, increases the prices of the medication products. The problem with these studies, however, lies in the complexity to identify and find relevant participants and to trust their participation. Hereby, there is a natural requirement for having highly signgificant results to protect the public from bad consequences of certain drugs.

Now, the medical wallet infrastructure allows pharmaceutical corporations to engage and approach the individual patients directly and also to reward them either directly or indirectly via the system providers, e.g., health insurers. For such participations there is also an extra wallet layer that, e.g., only shows the variations in raw data while taking the drug, such that patients may not reveal too much information to private research institutions. Further, pharmaceutical companies have the proof of the wallet existence via the system providers. This allows them to trust the wallet also on an anonymous basis of a hash value identifier, which makes also much easier for them to identify and approach a large group of patients to offer them a study participation. This has the aim to significantly reduce the time of setting up such studies and gaining significant results. As such, even a reduction of half a year in drug testing would mean an enormous cost reduction or increase in sales, respectively.

4 Medical Storage Infrastructure

IPFS Overview

The cost to store data on the Ethereum blockchain is designed to be prohibitive, for example storing 1GB would require 32000 Ether a cost of around $6,400000. In light of this the Inter Planetary File System a system for distributing and versioning large data is used to store the medical data. IPFS allows data to be easily propagated with

no single point of failure and the nodes making up the IPFS network do not need to trust each other

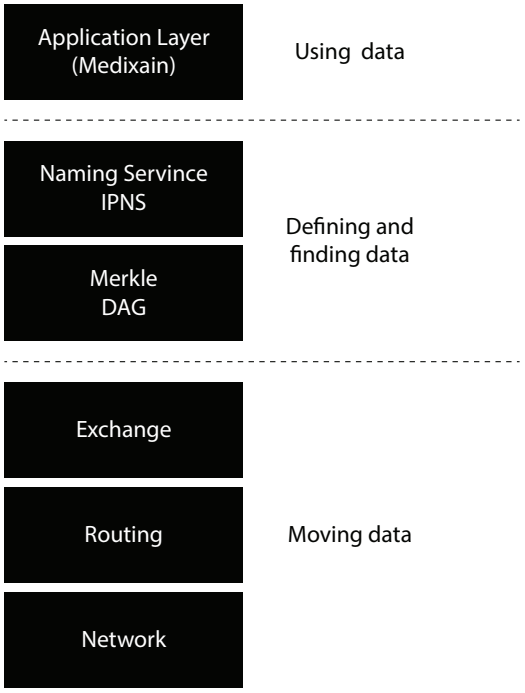

Figure 5: IPFS Stack

IPFS Stack IPFS uses a distributed sloppy hash table and block exchange mechanism to form a massive peer to peer system for storing and distributing blocks quickly and robustly. On top of this, IPFS builds a cryptographically authenticated data structure the Merkle directed acyclic graph (DAG) where links between objects are the cryptographic hashes of the targets. The Merkle DAG is used to model information structures such as file systems.

Because IPFS objects are contained in a Merkle DAG they can be

1. retrieved via their 512 bit hash
2. integrity checked
3. linked to other objects
4. cached indefinitely

This gives the IPFS system

1. Content Addressing: all content is addressed by a hash of the content

2. Tamper resistance: the content hash contains a checksum to prevent tampering or corruption

3. Deduplication: objects with the same content are only stored once.

The incentivised block exchange mechanism bitswap is a message based data trading module for IPFS, it manages requesting and sending blocks to and from other peers in the network. Indexing information allows particular content to be found efficiently and IPNS a naming service allows human readable names. The distributed nature of content delivery saves bandwidth and unlike the https protocol prevents DDoS attacks.

Similar Distributed Storage systems The main competitors to IPFS are Storj and Swarm. Storj is not suitable as it does not have a concept of a private network and storage is charged for. Swarm makes a better alternative, it is part of the Ethereum stack, and unlike IPFS does not require nodes to stay online. However Swarm is currently not ready for production. To allow medixain to switch to alternative storage systems as required, the application layer defines a storage abstraction, to allow alternative storage implementations to be plugged in.

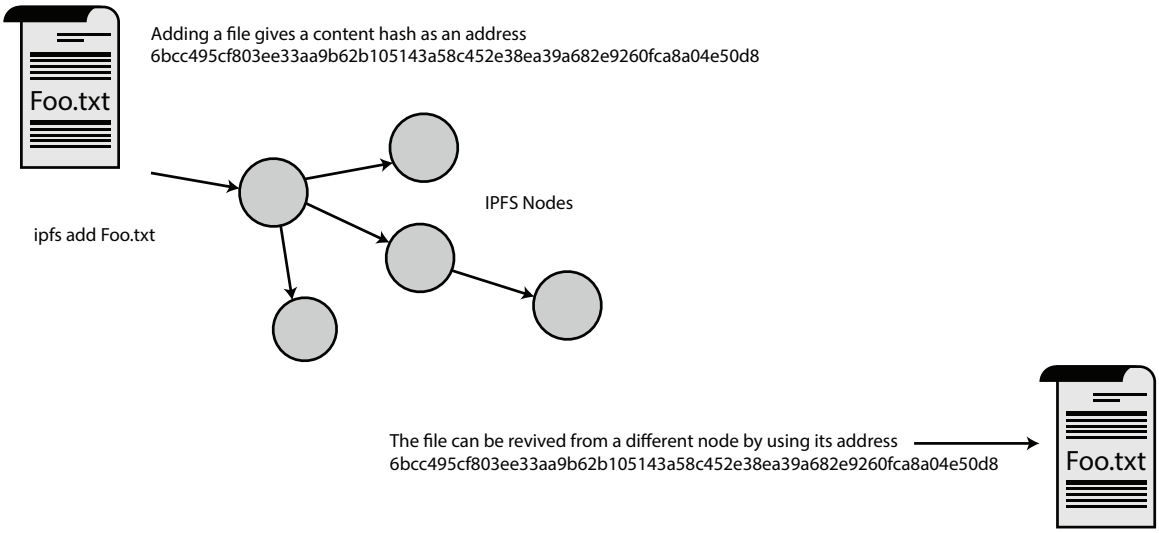

Figure 6: Data Addition Diagram

Data Addition Data is added to IPFS through client software that forms an IPFS node. the client exposes an API to add and retrieve data.

Data Deletion Files can be attached to nodes by a 'pinning' process,by unpinning files from a node, they are made available for local removal by the garbage collector. The process to remove data is then

1. unpin the file from any node it is pinned to.

2. make sure that no other modes request the file before it can be garbage collected,since we have control over the nodes in the network we can ensure this.

Proxy Re-Encryption Overview

Proxy re-encryption allows a proxy to re encrypt an existing ciphertext encrypted by one key, so that it may be decrypted by different key without learning anything about the underlying message. The goal of this is to securely enable re encryption without the need to rely on any trusted parties Without proxy re encryption the data owner needed to trust say a keyserver to act honestly and correctly, it also allowed the owner of key server access to all data. Existing key encryption models are suited to 1 to 1 messaging whereas proxy re encryption is more scalable for N-to-N communication. We need not know the recipient of the message in advance since this is specified when when create the re encryption token.[34]

Proxy re-encryption schemes add two functions to traditional encryption schemes:

1. Delegation, the data owner produces a re-encryption key based on his secret key and the key of a delegated user.

2. Transitivity, the original data can be re encrypted any number of times, allowing access to any number of delegates

Proxy re encryption has been shown not to weaken the underlying system [35]. The ability to re encrypt the data allows data sharing to be controlled by the owner of the data, typically the patient. Our implementation is based on NuCypher KMS is a decentralized Key Management System [33]. The process is summarized as:

For a cypher text C_A encrypted with secret key S_A, we can produce a re encryption key R_k for public key P_k B

$$R_k = rekey(S_A, P_k\ B)$$

then re-encrypt the cypertext

$$C_B = re - encrypt(R_k, C_A).$$

B can then decrypt it with their private key.

$$D_B = decrypt(S_B, C_B)$$

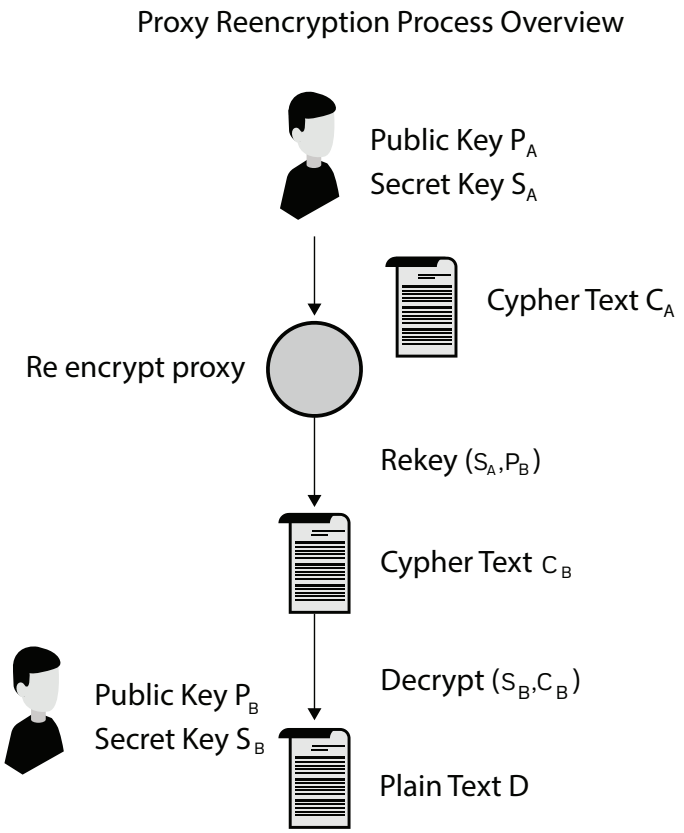

Figure 7: Proxy Re-Encryption Overview

The Ethereum Stack

Ethereum Virtual Machine overview The Ethereum virtual machine is a runtime environment for smart contracts, sandboxed and isolated from the network and host processes. Every node runs an EVM implementation. Once a transaction is initiated the whole state transition of the EVM is deterministic. This determinism is essential if consensus is to be achieved between the Ethereum nodes. A block can be seen as a unit of agreement between Ethereum nodes.

Data on the Ethereum blockchain is primarily held as the state of the EVM, the global EVM state, is a mapping between 160 bit Ethereum addresses and account states, these can be simple accounts where the status represents a currency value, or a smart contract where the state consists of program code and program state.

State transitions within the EVM are the result of transactions; For medixain the transaction T is defined by:

$$T \equiv (Tn, Tp, Tg, Tt, Tv, Td, Tw, Tr, Ts)$$

with

Tn: a nonce value incremented for each transaction an originating account performs. ($Tn \in \mathbb{P}\ 256$)

Tp: gas price, an amount to be paid in ether as a cost of computation steps. ($Tp \in \mathbb{P}\ 256$)

Tg: gas limit, effectively a limit on the amount of computation permitted for this transaction. ($Tg \in \mathbb{P}\ 256$)

Tt: message recipient for medixain transactions this is the address of the smart contract.

Tv: not used by medixain. ($Tv \in \mathbb{P}\ 256$)

Td: the transaction data, the function to be called along with its arguments. ($Td \in \mathbb{B}$)

Tw, Tr, Ts values associated with the ECDSA used to sign the transaction. ($Tw \in \mathbb{P}\ 5$, $Tr \in \mathbb{P}\ 256$, $Ts \in \mathbb{P}\ 256$)

Smart Contracts Medixain smart contracts are written using the solidity programming language, a Turing complete contract-oriented, high-level language that is the primary language used for Ethereum blockchains. It is chosen over the alternatives (serpent / viper / LLL) because of its relative ease of use, and its similarity to well known languages such as C and JavaScript. Smart contracts are general purpose units of code run within the EVM, they are restricted both to allow determinism (for example disallowing random numbers) and to prevent denial of service (using an economic disinsentive to discourage excessive computational steps).

Whisper a messaging protocol on Ethrerum Whisper is a cryptographic peer-to-peer network and protocol suite which provides a general-purpose transport and interface for applications to communicate via a p2p network. Use of whisper ensures that messages passed between Ethereum nodes or contracts are encrypted, and the message sender is be verifiable.

The Medixain Application

Medixain Private IPFS Network Medixain uses the IPFS client software to build a IPFS network that is made private by restricting node membership This is achieved by requiring nodes to have knowledge of a shared 512 bit key.This requirement is in addition to any firewall rules that would also restrict access to the private network.

Medixain IPFS Data Structures The medixain data structures are built upon the underlying IPFS structures. A medixain record R is defined as the 3 tuple

$$R \equiv (Kc, S, D)$$

where

Kc is a 256 bit symmetric encryption key further encrypted by a public key

S is the encryption scheme used (32 char)

D is an unlimited byte array of encrypted data

Furthermore, we define D as the following:

$$D \equiv (P, F, T, B)$$

where

P is a 160 bit patient identifier

F is a filename (64 char)

T is the file type (16 bit enum)

B is an unlimited byte array holding the file contents.

An abstraction layer interfaces between the medixain application and the IPFS API to control storage of the medixain records.

Within the medixain smart contract the state is represented in the solidity programming language as

```
struct\{

address originator

bytes IPFS

uint64 lastUpdate

\}
```

where

originator is the Ethereum address of the actor making the change in IPFS.

$IPFS$ is the IPFS address of the record.

lastUpdate is the unix time of the last update to the record.

Medixain use of Ethereum Events Events are special operations that produce an log like entry in the blockchain at less cost than storing data in the contract's state, this mechanism is often used to signal to external programs that a transaction has occurred. An event E is hereby the tuple $E \equiv (A, T, D)$ with

A as the originator Ethereum address $A \in \mathbb{P}\ 160$

T as a topic associated with numerous events $T \in \mathbb{P}\ 256$

D as the event data $D \in \mathbb{B}$

Privacy

A core feature of blockchains is transparency, this makes an applications privacy requirements challenging. Data on the blockchain is visible to anyone both as the current state of the EVM and as details of any transactions that produced that state. Typically data that is required to be private needs to be encrypted before it is added to the blockchain or IPFS. For auditability requirements only the hash of data needs to be on the blockchain, this also has privacy benefits since given the hash we are given no knowledge other than that data has been added and the Ethereum address of the originator.

Submitting Records Overview

Access to data is controlled re encryption nodes specified in the Key Management section. The medical professional and patient can collaborate in creating a data record R, which is submitted via client software to the IPFS network.

Key details The record is encrypted in the client with a random symmetric key K to produce cypher text C_A. The key K is then encrypted with the public key of the data owner (typically the patient) to produce K_c. K_c and C_A are then stored together on IPFS. A transaction recording this addition is then produced by the client software and submitted to the Ethereum network.

Our requirement for a secure history for medical records is achieved by the immutable addition of this transaction to the Ethereum blockchain.

Retrieving Records

This IPFS address of a record can be obtained from the Medixain smart contract, and thus the encrypted record can be obtained from IPFS.

This gives $R \equiv (Kc, S, D)$ as described above. The encryption will then proceed as

1. Retrieval by the record owner

   The owner can use their private key to decrypt the data key K_c, and use this key to decrypt the rest of the data record.

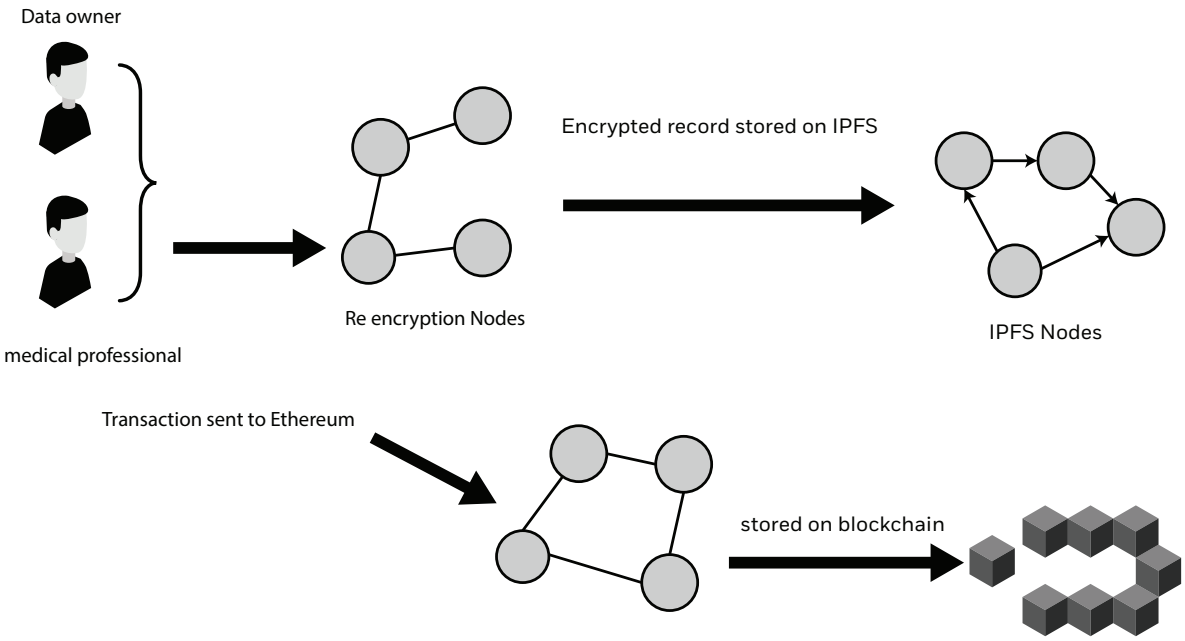

Figure 8: Overview of Adding Records

2. Retrieval by a third party

   To decrypt the record a re encryption node supplies a re enryption key K_r to any applicant allowed to view the data.

   K_r can be used to decrpyt the data key K_c, which can then be used to decrypt the data D.

   A transaction is sent from the re encryption node to the Medixain smart contract to record that access has been granted.

   If required access to a record on IPFS can be revoked by the re encryption nodes after a certain time.

5 Key Management

Although the use of cryptographic schemes allows efficient and secure message transfer, poor key management can lead for example to data becoming inaccessible if a key is lost, or confidential data being exposed if a key is compromised. The keys we will need to use are too long to be remembered by the users of our system. By the use of proxy re encryption we can effectively mitigate these problems, the system we use is based on NuCypher KMS a decentralized Key management system.

Key compromise or loss Since each record is encrypted by a different key, the compromise of this key would effect at most one record. Similarly the loss of that key

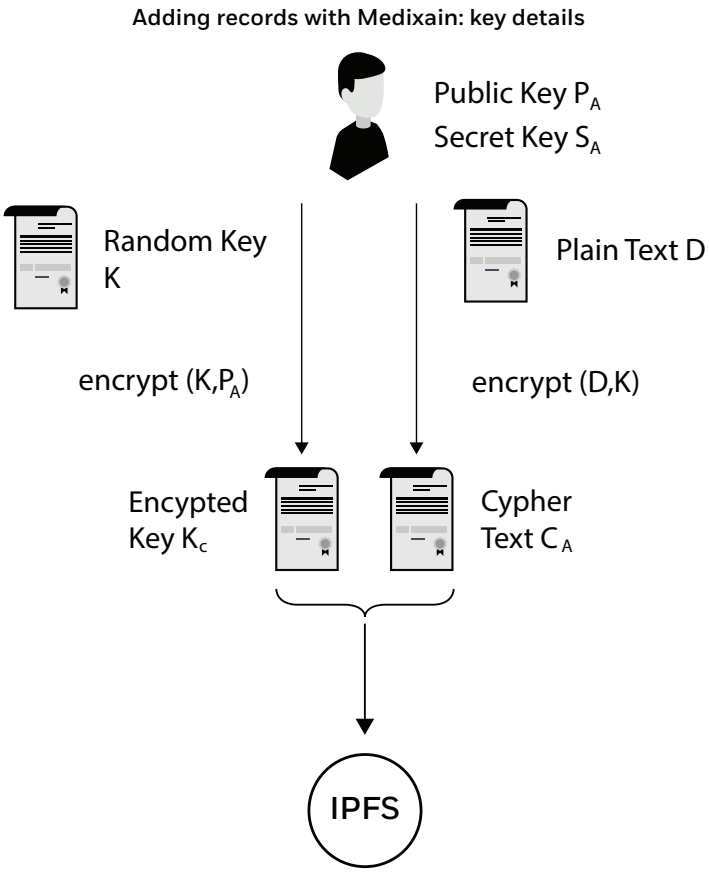

Figure 9: Overview of Adding Key Details

for make at most one record inaccessible.

Loss of user's public / private keys The loss or compromise of a patient's private key could have greater impact since it effects more records, to prevent this steps to allow regeneration of keys as detailed below should be taken.

Controlling access to IPFS data Controlling access to the data is handled by re encryption nodes, these use the existing nodes forming our private Ethereum network, to which specific programs interact to issue re encryption keys to users allowed to view the data.

The encryption node will receive a request S from a holder of public key P_s using the Whisper protocol to ensure request integrity.

$$S \equiv (P_s, K_c)$$

The re encryption node has a permissioning function $P\{P_s, K_c\} \rightarrow \{1, 0\}$ to allow or deny access for a public key. The user applies to a re encryption node for the relevant

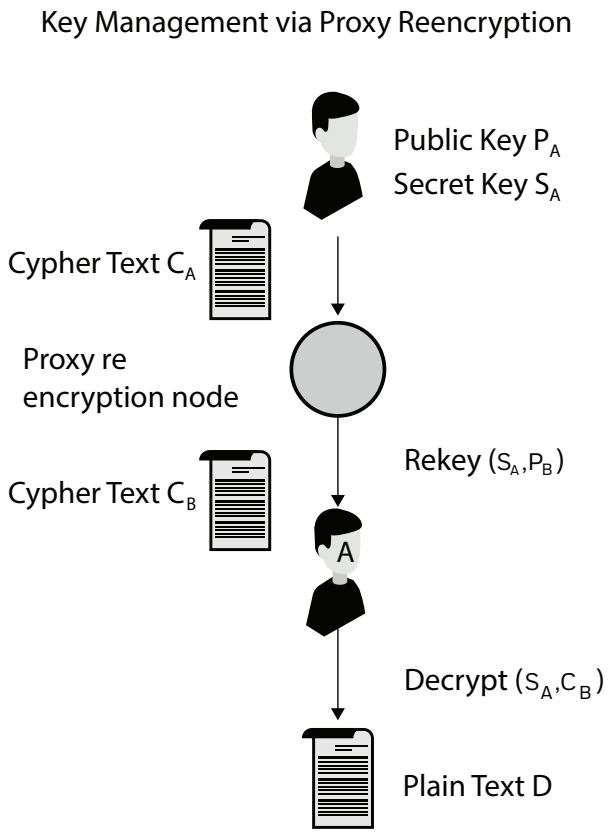

Figure 10: Key Management via Proxy Re-Encryption

key for an IPFS file. The re-encryption node will supply a re-encryption key from this the user can decrypt K_c to produce K, which can then be used to decrypt the data D.

The data owner can add or remove re-encryption keys to our proxy nodes, specifying which individuals or groups can access specific medical records, providing a dynamic layered system of access.

Regeneration of public keys The system is dependent on the continued access to the public / private key pair of the data owner. For ease of use Ethereum produced public / private keys would be used. For a given private key, the patient identifier is the 160 bit Ethereum address obtained by taking the rightmost 160 bits of the Keccak hash of the corresponding ECDSA public key. Standard techniques to preserve the private key could be used as appropriate, such as cold storage or use of a regeneration phrase.

Key revocation Data owners can revoke re-encryption keys by removing the from the proxy nodes, if either a key has been compromised, or the owner is implementing time based access.

6 Robust Consensus Optimization

6.1 Motivation

With the introduction and technical description of the actual application of the layered medical wallet system, we now also have the means to optimize the Blockchain network and particularly the underlying consensus mechanisms themselves. The reason for this clearly lies in the enormous sensitivity of medical data, which needs to persist possible attacks but also requires a certain network stability, which we may not get by using standard Blockchain architectures, such as the open Ethereum network.

As such, in the following we develop a new network – the Blockchain Intelligence Framework – which makes use of the corporate Ethereum structure, yet, defining an optimization framework with various tasks to fulfill: First of all, we allow for permissioning and network policies to integrate our defined KYC and access control layers for further verifications. This permssioning restricts the possible nodes and, thus, wallets in the network as well as the mining activity, which is highly crucial when dealing with highly sensitive data.

The consensus optimization play hereby the largest role and will be defined according to general assumptions that are integrated on protocol level. Having this knowledge we can build models that measure and influence the security, cost and stability of the entire system, in order to find an optimal equilibrium of the system state, based on the required parameters and lying within a certain monetary budget. This is particularly important, as open Proof of Work consensus systems are not fitting within any budget boundaries in the long term and other approaches, such as Proof of Stake and PBFT are clearly not suitable to function in systems that require a high stability and long term security of data.

Having having defined such an optimization model, we will use this functionality for advancements to also optimize the group dynamics of a fragmented infrastructure setting, such as when dealing with health insurers as the system and mining providers. These optimization frameworks will then inform the actual protocol level by defining machine learning engines that can compute and adapt the required network specifications at run time, such that we can have the stabilizing equilibrium effect, also in cases of unexpected failures of machines or possible intrusions. As such section **??** will model such engines, based on recent advances in reinforcement learning, yet, here the first time being applied and reengineered to permissioned Blockchain networks.

6.2 Mathematics for Centrally Governed Proof of Work

Our model assumes a cryptographic hash function

$$h\colon \{0,1\}^p \to \{0,1\}^n$$

where $p \geq n > 0$ such that h has *puzzle friendliness* [17]. The *level of difficulty* d is an integer satisfying $0 < d < n$: Proof of Work has to produce some x where $h(x)$ has

at least d many leftmost 0 bits. We write $T > 0$ for the time to compute a sole hash $h(x)$ and to decide whether it has at least d leftmost zeros. Since the range of d will be relatively small, we make T a device-dependent constant.

Our probabilistic modeling will treat h in the *Random Oracle Model* (ROM): function h is chosen uniformly at random from all functions of type $\{0,1\}^p \to \{0,1\}^n$; that is to say, h is a deterministic function such that any x for which h has not yet been queried will have the property that $h(x)$ is governed by a truly random probability distribution over $\{0,1\}^n$.

We may assume that x consists of a block header which contains some random data field – a nonce *nonce* of bitlength r, that this nonce is initialized, and that the nonce is then increased by 1 each time the hash of x does not obtain Proof of Work. In particular, this yields that $\{0,1\}^p \cong \{0,1\}^{p-r} \times \{0,1\}^r$ where $0 < r < p$: the input to h will be of form $x = data \mid\mid nonce$ where *data* and *nonce* have $p - r$ and r bits, respectively. Our use of ROM will rely on the following assumption:

Assumption 1 (Invariant) *The mining of a block with one or more miners will use an input to h at most once, be it within or across miners' input spaces.*

This assumption and ROM give us that hash values are always uniformly distributed in the output space during a mining race. Now consider having $s > 1$ many miners that run in parallel to find Proof of Work, engaging thus in a *mining race*. We assume these miners run with the same configurations and hardware. As already discussed, in our approach miners do not get rewarded:

Assumption 2 (Miners) *Miners are a resource controlled by the governing organization or consortium, and have identical hardware. In particular, miners are not rewarded nor have the need for incentive structures.*

But miners may be corrupted and misbehave, for example they may refuse to mine. To simplify our analysis, we assume miners begin the computation of hashes in approximate synchrony:

Assumption 3 (Approximate Synchrony) *Miners start a mining race at approximately the same time.*

For many application domains, this is a realistic assumption as communication delays to miners would have a known upper bound that our models could additionally reflect if needed.

Next, we want to model the *race* of getting a Proof of Work where each miner j has some data $data_j$. To realize Assumption 1, it suffices that each miner j has a nonce $nonce_j$ in a value space of size

$$\lambda = \lfloor 2^r / s \rfloor$$

such that these nonce spaces are mutually disjoint across miners.

Our probability space has $(data_j)_{1\leq j\leq s}$ and d as implicit parameters. For each miner j, the set of basic events E^j is

$$E^j = \{\otimes^k \cdot \checkmark \mid 0 \leq k \leq \lambda\} \cup \{\mathsf{failure}\} \tag{1}$$

Basic event failure denotes the event that all λ nonce values $nonce_j$ from $\{(j-1)\cdot\lambda, \ldots, j\cdot\lambda - 1\}$ for miner j failed to obtain Proof of Work for $data_j$ at level of difficulty d. Basic event $\otimes^k \cdot \checkmark$ models the event in which the first k such nonce values failed to obtain Proof of Work for $data_j$ at level d but the $k+1$th value of $nonce_j$ did render such Proof of Work for $data_j$.

To model this mining race between s miners for $(data_1, data_2, ..., data_s)$ and d as implicit parameters, we take the product $\prod_{j=1}^s E^j$ of s copies E^j and quotient it via an equivalence relation $\equiv$ on that product $\prod_{j=1}^s E^j$, which we now define formally.

Definition 1 *1. The s-tuple* $(\mathsf{failure}, \ldots, \mathsf{failure})$ *models failure of this mining race, and it is* $\equiv$ *equivalent only to itself.*

2. All s-tuples $a = (a_j)_{1\leq j\leq s}$ *other than tuple* $(\mathsf{failure}, \ldots, \mathsf{failure})$ *model that the mining race succeeded for at least one miner. For such an s-tuple a, the set of natural numbers k such that* $\otimes^k \cdot \checkmark$ *is a coordinate in a is non-empty and therefore has a minimum* $\min(a)$*. Given two s-tuples* $a = (a_j)_{1\leq j\leq s}$ *and* $b = (b_j)_{1\leq j\leq s}$ *both different from* $(\mathsf{failure}, \ldots, \mathsf{failure})$*, we can then define*

$$a \equiv b \text{ iff } \min(a) = \min(b)$$

So two non-failing tuples are equivalent if they determine a first (and so final) Proof of Work at the same round of the race. This defines an equivalence relation $\equiv$ and adequately models a synchronized mining race between s miners.

The interpretation of events $\otimes^k \cdot \checkmark$ in the mining race is then the equivalence class of all those tuples a for which $\min(a)$ is well defined and equals k: all mining races that succeed first at round k. The meaning of failure is still overall failure of the mining race, the equivalence class containing only tuple $(\mathsf{failure}, \ldots, \mathsf{failure})$. The set of events for the Proof of Work race of s miners is therefore

$$E^s = \{\otimes^k \cdot \checkmark \mid 0 \leq k \leq \lambda\} \cup \{\mathsf{failure}\} \tag{2}$$

In (2), expression $\otimes^k \cdot \checkmark$ denotes an element of the quotient

$$\Big(\prod_{j=1}^{s} E^j\Big)/\equiv$$

namely the equivalence class of tuple $(\otimes^k \cdot \checkmark, \mathsf{failure}, \mathsf{failure}, \ldots, \mathsf{failure})$. Next, we define a probability distribution $prob^s$ over E^s. To derive the probability $prob^s(\otimes^k \cdot \checkmark)$, recall

$$\tilde{p}(\otimes^k) = (1 - 2^{-d})^k$$

as the probability that a given miner does not obtain Proof of Work at level d in the first k rounds. By Assumption 1, these miners work independently and over disjoint input spaces. By ROM, the expression

$$\left[(1-2^{-d})^k\right]^s = (1-2^{-d})^{k\cdot s}$$

therefore models the probability that none of the s miners obtains Proof of Work in the first k rounds. Appealing again to ROM and Assumption 1, the behavior at round $k+1$ is independent of that of the first k rounds. Therefore, we need to multiply the above probability with the one for which at least one of the s miners will obtain a Proof of Work in a single round. The latter probability is the complementary one of the probability that none of the s miners will get a Proof of Work in a sole round, which is $(1-2^{-d})^s$ due to the ROM independence. Therefore, we get

$$prob^s(\otimes^k \cdot \checkmark) = (1-2^{-d})^{k\cdot s} \cdot [1-(1-2^{-d})^s] \tag{3}$$

This defines a probability distribution with a non-zero probability of $\mathsf{failure}$. Firstly,

$$\sum_{k=0}^{\lambda} (1-2^{-d})^{k\cdot s} \cdot [1-(1-2^{-d})^s]$$

is in $(0,1)$: to see this, note that this sum equals

$$[1-(1-2^{-d})^s] \cdot \frac{1-[(1-2^{-d})^s]^{\lambda+1}}{1-(1-2^{-d})^s} = 1-(1-2^{-d})^{s\cdot(\lambda+1)}$$

Since $0 < d, s$, the real $1-2^{-d}$ is in $(0,1)$, and the same is true of any integral power thereof. Secondly, $prob^s$ becomes a probability distribution with the non-zero probability $prob^s(\mathsf{failure})$ being $1 - prob^e(E^s \setminus \{\mathsf{failure}\})$, that is

$$prob^s(\mathsf{failure}) = (1-2^{-d})^{s\cdot(\lambda+1)} \tag{4}$$

That this failure probability is almost identical to that for $s = 1$ is an artefact of our modeling: if each miner has 64 bits of nonce space, e.g., then our model would have $r = 64 \cdot s$, so failure probabilities do decrease as s increases.

6.3 Mathematical Optimization in Mining Design Space

Generality of Approach We want to optimize the use of $s > 1$ miners using a level of difficulty d, and a bit size r of the global nonce space with respect to an objective function. The latter may be a cost function, if containing cost is the paramount objective or if a first cost estimate is sought that can then be transformed into a constraint to optimize for a security objective – for example to maximize the level of difficulty d, as seen further below. Higher values of d add more security: it takes more effort to mine

$$
\begin{array}{rcl}
0 & < & s_l \leq s \leq s_u \qquad 0 < d_l \leq d \leq d_u \qquad 0 < r_l \leq r \leq r_u \qquad \epsilon \geq prob^s(\mathsf{failure}) \\
\tau_u & \geq & T \cdot E^s(noR) \geq \tau_l \qquad\qquad \delta_2 \geq prob^s(disputes\ within\ \mu) \\
\delta & \geq & prob^s(PoWTime > th) \qquad \delta_1 \geq prob^s(PoWTime < th')
\end{array}
$$

Figure 11: Constraint set $\mathcal{C}$ for two optimization problems: (a) *minimize* $\mathrm{Cost}(s, r, d)$ as in (5) subject to constraints in $\mathcal{C}$; and (b) *maximize* d subject to $\mathcal{C} \cup \{\mathrm{Cost}(s, r, d) \leq budget\}$ for cost bound *budget*. This is parameterized by constants $0 \leq \delta, \delta_1, \delta_2, \epsilon, th, th', \tau_l, \mathrm{TVC}, \mathrm{TFC}$ and $0 < T, s_l, r_l, d_l$. Variables or constants $s_l, s_u, s, d_l, d_u, d, r_l, r_u, r$ are integral

a block and so more effort to manipulate the mining process and used consensus mechanism. But lower values of d may be needed, for example, in high-frequency trading where performance can become a real issue. We want to understand such trade-offs. Moreover, we want to explore how corruption of some miners or inherent uncertainty in the number of deployed miners or in the level of difficulty across the lifetime of a system may influence the above tradeoffs.

Optimizing Cost and Security The flexibility of our approach includes the choice of an objective function for optimization. Let us first consider an objective function

$$\mathrm{Cost}(s, r, d) = \mathrm{TVC} \cdot E^s(noR) \cdot s + \mathrm{TFC} \cdot s \tag{5}$$

that models cost as a function of the number of miners s, the bit size of the nonce r – implicit in random variable $E^s(noR)$, and the level of difficulty d; where we want to *minimize* cost.

The real variable TVC models the *variable* cost of computing *one* hash for *one* miner, reflecting the device-dependent speed of hashes and the price of energy. The real variable TFC models the *fixed* costs of *having one miner*; this can be seen as modeling procurement and depreciations. Variables s, r, and d are integral, making this a *mixed integer* optimization problem [10]. The expression $E^s(noR)$ denotes the *expected number of rounds* (of approximately synchronous hash attempts) needed to mine a block in a mining race that uses s miners, level of difficulty d, and nonce bitsize r. The derivation of this expression below shows that it is non-linear, making this a MINLP optimization problem [19, 10].

We may of course use other objective functions. One of these is simply the expression d, which we would seek to *maximize*, the intuition being that higher values of d give us more trust into the veracity of a mined block and the blockchains generated in the system. Figure 11 shows an example of a set of constraints and optimizations of security and cost for this.

Integer constants s_l and s_u provide bounds for variable s, and similar integer bounds are used to constrain integer variables r and d. The constraint for ϵ uses it as upper bound for the probability of a mining race failing to mine a block. The next two

inequalities stipulate that the expected time for mining a block is within a given time interval, specified by real constants τ_l and τ_u.

The real constant δ_2 is an upper bound for

$$prob^s(disputes\ within\ \mu)$$

the probability that more than one miner finds PoW within μ seconds in the same, approximately synchronous, mining race. The constraint for real constant δ says that the probability

$$prob^s(PoWTime > th)$$

of the *actual* time for mining a block being above a real constant th is bounded above by δ. This constraint is of independent interest: knowing that the expected time to mine a block is within specified bounds may not suffice in systems that need to assure that blocks are *almost always* (with probability at least $1 - \delta$) mined within a specified time limit. Some systems may also need assurance that blocks are almost always mined in time *exceeding* a specified time limit th'. We write

$$prob^s(PoWTime < th')$$

to denote that probability, and add a dual constraint, that the actual time for mining a block has a sufficiently small probability $\leq \delta_1$ of being faster than threshold th'.

Constraints as Analytical Expressions We derive analytical expressions for random variables occurring in Figure 11. Beginning with $E^s(noR)$, we have

$$E^s(noR) = \sum_{0 \leq k \leq \lambda} prob^s(\otimes^k \cdot \checkmark) \cdot (k+1) \tag{6}$$

which we know to be equal to

$$\sum_{0 \leq k \leq \lambda} (1 - 2^{-d})^{k \cdot s} \cdot [1 - (1 - 2^{-d})^s] \cdot (k+1)$$

We may rewrite the latter expression so that summations are eliminated and reduced to exponentiations: concretely, we rewrite $\sum_{0 \leq k \leq \lambda} prob(\otimes^k \cdot \checkmark) \cdot (k+1)$, the righthand side of (6), to $\lambda + 1$ summations, each one starting at a value between 0 and λ, where we exploit the familiar formula

$$\sum_{k=a}^{b} x^k = \frac{x^a - x^{b+1}}{1 - x}$$

This renders

$$E^s(noR) = \frac{1 - y^{\lambda+1} - (\lambda+1) \cdot (1-y) \cdot y^{\lambda+1}}{1 - y} \tag{7}$$

where we use the abbreviation

$$y = (1 - 2^{-d})^s \tag{8}$$

The expected time needed to get a proof of work for input *data* is then given by

$$E^s(poW) = T \cdot E^s(noR) \tag{9}$$

We derive an analytical expression for the probability $prob^s(PoWTime > th)$ next. Note that $(th/T) - 1 < k$ models that the actual time taken for $k+1$ hash rounds is larger than th. Therefore, we capture $prob^s(PoWTime > th)$ as

$$\sum_{\lceil (th/T)-1 \rceil < k \leq \lambda} prob^s(\otimes^k \cdot \checkmark) = y^{\lceil (th/T)-1 \rceil + 1} - y^{\lambda+1} \tag{10}$$

assuming that $\lceil (th/T) - 1 \rceil < \lambda$, the latter therefore becoming a constraint that we need to add to our optimization problem. One may be tempted to choose the value of δ based on the Markov inequality, which gives us

$$prob^s(PoWTime \geq th) \leq T \cdot E^s(noR)/th$$

But we should keep in mind that upper bound $T \cdot E^s(noR)/th$ depends on the parameters s, r, and d; for example, the analytical expression for $E^s(noR)$ in (7) is dependent on λ and so dependent on r as well. The representation in (10) also maintains that expression $y^{\lceil (th/T)-1 \rceil + 1} - y^{\lambda+1}$ is in $[0, 1]$, i.e. a proper probability. Since $y = (1 - 2^{-d})^s$ is in $(0, 1)$, this is already guaranteed if $\lceil (th/T) - 1 \rceil + 1 \leq \lambda + 1$, i.e. if $\lceil (th/T) - 1 \rceil \leq \lambda$. But we already added that constraint to our model. Similarly to our analysis of $prob^s(PoWTime > th)$, we get

$$prob^s(PowTime < th') = 1 - (1 - 2^{-d})^{s \cdot (\lfloor (th'/T)-1 \rfloor + 1)} = 1 - y^{\lfloor (th'/T)-1 \rfloor + 1} \tag{11}$$

which needs $0 < \lfloor (th'/T) - 1 \rfloor$ as additional constraint.

To derive an analytical expression for $prob^s(disputes\ within\ \mu)$, each miner can perform $\lfloor \mu/T \rfloor$ hashes within μ seconds. Let us set

$$w = (1 - 2^{-d})^{\lfloor \mu/T \rfloor + 1} \tag{12}$$

The probability that a given miner finds PoW within μ seconds is

$$\sum_{k=0}^{\lfloor \mu/T \rfloor} (1 - 2^{-d})^k \cdot 2^{-d} = 2^{-d} \cdot \frac{1 - (1 - 2^{-d})^{\lfloor \mu/T \rfloor + 1}}{1 - (1 - 2^{-d})} = 1 - w \tag{13}$$

Therefore, the probability that no miner finds PoW within μ seconds is

$$prob^s(0\ PoW\ within\ \mu) = (1 - (1 - w))^s = w^s \tag{14}$$

The probability that exactly one miner finds PoW within μ seconds is

$$prob^s(1\ PoW\ within\ \mu) = s \cdot w^{s-1} \cdot (1 - w) \tag{15}$$

$$
\begin{array}{rcl}
s_l & \leq & s \leq s_u \qquad d_l \leq d \leq d_u \qquad r_l \leq r \leq r_u \qquad \lambda = \lfloor 2^r/s \rfloor \\
y & = & (1-2^{-d})^s \qquad w = (1-2^{-d})^{\lfloor \mu/T \rfloor + 1} \qquad 0 \leq \lfloor \mu/T \rfloor \\
\epsilon & \geq & y^{\lambda+1} \qquad \lceil (th/T) - 1 \rceil < \lambda \qquad 0 < \lfloor (th'/T) - 1 \rfloor \\
E^s(noR) & = & \dfrac{1 - y^{\lambda+1} - (\lambda+1) \cdot (1-y) \cdot y^{\lambda+1}}{1-y} \\
\tau_u & \geq & T \cdot E^s(noR) \geq \tau_l \qquad \delta_1 \geq 1 - y^{\lfloor (th'/T) - 1 \rfloor + 1} \\
\delta & \geq & y^{\lceil (th/T) - 1 \rceil + 1} - y^{\lambda+1} \\
\delta_2 & \geq & 1 + (s-1) \cdot w^s - s \cdot w^{s-1}
\end{array}
\tag{17}
$$

Figure 12: Arithmetic version of set of constraints $\mathcal{C}$ from Figure 11, with additional soundness constraints for this representation. Feasibility of (s, r, d) and $r_u \geq r' > r$ won't generally imply feasibility of (s, r', d) due to the constraint in (17)

Thus, the probability that more than one miner finds PoW within μ seconds is

$$
\begin{array}{rcl}
prob^s(disputes\ within\ \mu) & = & 1 - prob^s(0\ PoW\ within\ \mu) - prob^s(1\ PoW\ within\ \mu) \\
& = & 1 - w^s - s \cdot w^{s-1} \cdot (1-w) \\
& = & 1 - w^s - s \cdot w^{s-1} + s \cdot w^{s-1} \cdot w \\
& = & 1 + (s-1) \cdot w^s - s \cdot w^{s-1}
\end{array}
\tag{16}
$$

Figure 12 shows the set of constraints $\mathcal{C}$ from Figure 11 with analytical expressions and their additional constraints, we add constraint $0 \leq \lfloor \mu/T \rfloor$ to get consistency for the analytical representation of $prob^s(disputes\ within\ \mu)$.

6.4 Robust Design Security

Our model above captures design requirements or design decisions as a set of constraints, to optimize or trade off measures of interest subject to such constraints. We can extend this model to also *manage uncertainty* via robust optimization [2]. Such uncertainty may arise during the lifetime of a system through the possibility of having corrupted miners, needed flexibility in adjusting the level of difficulty, and so forth. For example, corrupted miners may refuse to mine, deny their service by returning invalid block headers, pool their mining power to get more mining influence or they may simply break down. Robust optimization treats such uncertain as non-deterministic choice and refers to it as *strict* or *Knightian* uncertainty.

Consider $1 \leq l < s$ corrupted miners. We can model their *pool power* by appeal to ROM and the fact that the mining race is approximately synchronized: the probability that these l miners win $c > 0$ many subsequent mining races is then seen to be $(l/s)^c$. We can therefore bound this with a constant δ_3 as in Figure 12.

We model uncertainty in the number of miners available by an integer constant u_s as follows: if s miners are deployed, then we assume that at least $s - u_s$ and at most s many miners participate reliably in the mining of legitimate blocks: they will not mine blocks that won't verify and only submit mined blocks that do verify to the network. Constant u_s can model aspects such as denial of service attacks or a combination of such attacks with faults: $u_s = 3$, e.g., subsumes the scenario in which one miner fails and two miners mine invalid blocks.

Integer constant u_d models the uncertainty in the deployed level of difficulty d: intuitively, our analysis should give us results that are robust in that they hedge against the fact that any of the values d' satisfying

$$|\, d - d' \,| \leq u_d$$

may be the actually running level of difficulty. This enables us to understand a design if we are unsure about which level of difficulty will be deployed or if we want some flexibility in dynamically adjusting the value of d in the running system.

The corresponding robust optimization problem for cost minimization is seen in Figure 13. It adds to the constraints we already consider further requirements on constants l, c, and δ_3 as well as the constraint

$$l^c \leq \delta_3 \cdot s^c$$

The robustness of analysis is achieved by a change of the objective function from $\mathrm{Cost}(s, r, d)$ to

$$\mathrm{Cost}^{u_s}_{u_d}(s, r, d) = \max_{s - u_s \leq s' \leq s,\, |d - d'| \leq u_d} \mathrm{Cost}(s', r, d') \tag{18}$$

The latter computes a worst-case cost for triple (s, r, d) where s and d may vary independently subject to the strict uncertainties u_s and u_d, respectively. We call a triple (s, r, d) *feasible* if it satisfies all constraints of its optimization problem. Costs such as the one in (18) for a triple (s, r, d) are only considered for optimization if all triples (s', r, d') used in (18) are feasible – realized with predicate $\mathit{feasible}^{u_s}_{u_d}$: robust optimization guarantees [2] that the feasibility of solutions is invariant under the specified strict uncertainty (here u_s and u_d).

7 Distributed Institutional Mining Architecture

In this section, we study how our mathematical model for Proof of Work can be adjusted to deal with a finite number of types of miners, where each type has a different hash rate, cost or both. The motovation for this adaption is particularly the scenario of having a consortium of various institutions that share the network itself for different applications but have the requirement to integrate or share different computational inputs. This is a as such a game theoretical problem, as each institution has either the optimality condition to minimize its spending on the system or to maximize its computational

$\min\{\text{Cost}^{u_s}_{u_d}(s,r,d) \mid \mathit{feasible}^{u_s}_{u_d}(s,r,d)\}$
subject to the set of constraints $\mathcal{C}$ from Figure 12 together with

$4 = l < s$	$c = 6$	$0.001 = \delta_3$
$l^c \leq s^c \cdot \delta_3$	$u_s = 5$	$u_d = 3$

Figure 13: Robust cost optimization for the set of constraints from Figure 12, where up to $u_s = 5$ miners may be non-functioning, refusing to mine or mining invalid blocks; where the level of difficulty may vary by up to $+/-3$; and where the probability of any mining *pool* of size $l = 4$ winning $c = 6$ consecutive mining races is sufficiently small (here $\delta_3 = 0.001$). Predicate $\mathit{feasible}^{u_s}_{u_d}(s,r,d)$ characterizes *robustly feasible* triples and is true iff all triples (s',r,d') with $s - u_s \leq s' \leq s$ and $|d - d'| \leq u_d$ are feasible

influence on the system to dominate the decisions. Both scenarios, however, lead to a system crush, such that we need to find optimal policies and models to actually achieve a positive overall outcome. Further, we want to use this model to analyze or better understand a number of scenarios, including the following:

- How can or should we adapt the Proof of Work system when the overall system is suddenly under attack?
- How would our model behave (in reality but also in terms of mathematical analysis) if further nodes and servers were to be added to production?
- If a consortium wishes to provide a Proof of Work service collaboratively, how should they resource this given their individual resources and constraints, so that Proof of Work has sufficient resiliency to manipulation from within the consortium?

We note that the last scenario may also have to reflect the questions asked in the previous two scenarios. Let us recall the current assumptions of our mathematical model for Proof of Work:

A1 All servers hash at the same rate T.

A2 All servers have the same costs TVC and TFC.

A3 The uncertainty in the level of difficulty is u_d

A4 The number of corrupted or failing servers is less than or equal to u_s.

A5 It is specified that the probability of l corrupted miners winning c consecutive mining races is sufficiently small.

7.1 Heterogenous Groups of Mining Units

We now assume that we have $g > 1$ groups of miners, where all miners within a group have the same hash rate and cost, and where miners from different groups have different hash rate, different cost or both. We stress that these are *physical* groups of miners that have the same specifications, not *logical* groups such as the miners under the control of a particular partner in a consortium.

We will now provide some examples of how such physical groups might map onto scenarios of interest.

Example 1 *Here are some salient examples in which physical groups of miners are of interest:*

- *Let g equal 2 with physical group 1 comprising the miners already deployed and physical group 2 as the set of new miners at higher hash rates that could be added to the system.*
- *A variant of that scenario is when g equals 2 and we are also prepared to decommission miners from physical group 1 to get optimal balance between miners with two types of hash rates, where miners from physical group 2 have higher hash rates.*
- *Consider the case when g equals 4 but where we also have a consortium of 3 organizations (three logical groups) that each provide part of the Proof of Work service. We then want to understand how the resources from the 4 physical groups of miners are best mapped onto the 3 consortium members so that the service provides key performance and security characteristics.*

Subsequently, we will use the terms "group" and "type" interchangeably when referring to sets of homogeneous mining units.

7.2 Constants and Variables for Miner Types

We write T_i for the hash rate of miners of physical type i, TVC_i and TFC_i for their respective cost factors, and s_i for the number of miners of physical type i that will be deployed in the system. For each physical type i, we have constants s_i^l and s_i^u that serve as lower and upper bounds, respectively, on how many miners of physical type i can be deployed. It makes sense to also have constants s_l and s_u that are lower, respectively, upper bounds for how many miners s *across all physical types* can be deployed in the overall system in total. We therefore have the following constraints:

$$s_i^l \leq s_i \leq s_i^u \qquad (\forall 1 \leq i \leq g) \tag{19}$$

$$s_l \leq s \leq s_u \tag{20}$$

$$s = \textstyle\sum_{i=1}^{g} s_i \tag{21}$$

$$s_i^l, s_i^u, s_l, s_u \text{ non-negative integer constants} \tag{22}$$

$$s_i, s \text{ integer variables} \tag{23}$$

As before, we have a level of difficulty d and constants that bound that:

$$d_l \leq d \leq d_u \tag{24}$$

$$d_l, d_u \text{ non-negative integer constants} \tag{25}$$

$$d \text{ integer variable} \tag{26}$$

In particular, this assumes that all physical types of miners are faced with the same level of difficulty. When we model the nonce space, there is a question of whether this nonce space would be local to miners of each physical or logical type (so that miners of different physical or logical type may use the same nonce values) or whether it would be global so that miners across all physical and logical types have a unique local nonce space. The latter makes for a simpler model, whereas the former may accommodate implementations better, for example when consortium partners (i.e. logical types) would not like to manage consistency of such a global nonce space.

Our model will take the former approach here: it has a simpler model and its analysis results may approximate well systems that implement local nonce spaces. We have constants r_l and r_u as before with an integer variable r, representing the number of bits of the global nonce space:

$$r_l \leq r \leq r_u \tag{27}$$

$$\lambda = \lfloor 2^r / s \rfloor \tag{28}$$

$$r_l, r_u \text{ non-negative integer constants} \tag{29}$$

$$r \text{ integer variable} \tag{30}$$

Note that this considers that there is a totality of s miners across all physical types, and each one of them gets the same share size of the nonce space as in our original model when g was 1.

Let us also redefine the cost function to this group-based setting:

$$Cost(s, r, d) = \sum_{i=1}^{g} \Big(\mathrm{TVC}_i \cdot (\min_{j=1}^{g} E^{s_j}(noR)) \cdot s_i + \mathrm{TFC}_i \cdot s_i \Big) \tag{31}$$

This generalizes the cost function of our original model by summing costs over all physical groups i. The term $E^{s_j}(noR)$ represents the expected number of rounds of hashes for the mining race within physical group j. Therefore, the term $\min_{j=1}^{g} E^{s_j}(noR)$ is a conservative upper bound on the expected number of rounds of hashes for mining races across all g physical groups. The derivation of the group-specific probabilities $E^{s_j}(noR)$ and the argument for why this minimum is a conservative upper bound of the expected number of rounds for the entire system are given below.

7.3 Probability Spaces for Group-Based Model

It seems to be difficult to define a probability space for all s miners for the random variables of our model. For example, a formal derivation of the expected time to mine for

s miners with different hash rates is non-obvious and seems to require reasoning about discrete time points and the relativeness of different hash rates. We here therefore take a more abstract view, and define a probability space for each physical type of miner. This simplifies the model, but it also means that we have to be careful so that these abstractions don't make constraints of our model unsound.

We define

$$E^{s_i} = \{\otimes^k \cdot \checkmark \mid 0 \leq k \leq \lambda\} \cup \{\mathsf{failure}\} \tag{32}$$

which looks the same as our original E^s in terms of events but these events will have probabilities pertaining to physical group i. For each physical group i, we therefore consider s_i-tuples a and b and define the equivalence relation $\equiv_i$ as before for $\equiv$ but now over tuples of size s_i, one entry per miner in physical group i. This gives us an event space as the quotient $E^{s_i}/\equiv_i$ with representatives of equivalence classes as before: the s_i-tuple that contains only $\mathsf{failure}$ for the class of failures to mine, and the s_i-tuple $(\otimes^k \cdot \checkmark, \mathsf{failure}, \ldots, \mathsf{failure})$ for the class that finds Proof of Work at round k for the mining race confined to members of group i.

For group i, we have that $(1-2^{-d})^{k \cdot s_i}$ models the probability that none of the s_i miners in physical group i obtains Proof of Work in the first k rounds. From this we get that

$$prob^{s_i}(\otimes^k \cdot \checkmark) = (1-2^{-d})^{k \cdot s_i} \cdot [1-(1-2^{-d})^{s_i}] \tag{33}$$

is the probability of physical group i finding Proof of Work at round k. Since $0 < d, s_i$ this does define a probability distribution over E^{s_i} where we have that the probability of failure is

$$prob^{s_i}(\mathsf{failure}) = (1-2^{-d})^{s_i \cdot (\lambda+1)} \tag{34}$$

We therefore approximate a specification that the overall probability of failure is no larger than ϵ by

$$\epsilon \geq \min\{y_i^{\lambda+1} \mid 1 \leq i \leq g\} \tag{35}$$

where we set

$$y_i = (1-2^{-d})^{s_i} \tag{36}$$

Note that we did use the min operator here, since it suffices to know that the smallest failure probability for all i mining races within the i physical groups is not larger than ϵ. The joint failure probability for the sole, concurrent mining race across all groups cannot be larger than that minimum.

7.4 Random Variables in Group-Based Model

The computation of the expected number of rounds for physical group i to find Proof of Work is similar to our original model, and we get that

$$E^{s_i}(noR) = \sum_{0 \leq k \leq \lambda} prob^{s_i}(\otimes^k \cdot \checkmark) \cdot (k+1) \tag{37}$$

$$= \frac{1 - y_i^{\lambda+1} - (\lambda+1) \cdot (1-y_i) \cdot y_i^{\lambda+1}}{1-y_i} \tag{38}$$

The expected time to get Proof of Work on *data* for physical group i is therefore

$$E^{s_i}(poW) = T_i \cdot E^{s_i}(noR) \tag{39}$$

What constraints should we derive now about the expected time to mine for the overall system, given these expected times for each physical group i? First of all, in the original model we did not really use τ_l and so we may drop this here as well; doing this should make it easier to get a sound abstraction of the overall expected time to find Proof of Work. We propose to use the constraints

$$\begin{aligned} E^{s_i}(noR) &= \frac{1 - y_i^{\lambda+1} - (\lambda+1)\cdot(1-y_i)\cdot y_i^{\lambda+1}}{1-y_i} \qquad (\forall 1 \leq i \leq g) && (40)\\ \tau_u &\geq \min_{i=1}^{g}\Big\{T_i \cdot E^{s_i}(noR)\Big\} && (41)\\ & \quad \tau_u \text{ real-valued positive constant} && (42) \end{aligned}$$

We emphasize that the use of $\min_{i=1}^{g}\Big\{T_i \cdot E^{s_i}(noR)\Big\}$ is an abstraction that is sound in the context of an upper bound τ_u. Each physical group i participates in the same weakly asynchronous mining race. So whichever physical group i_0 has the smallest average time $t_0 = T_{i_0} \cdot E^{s_{i_0}}(noR)$ to find Proof of Work, this offers a guarantee that the expected value of the overall system to produce Proof of Work cannot be *larger* than that number t_0. So the truth of constraint $\tau_u \geq t_0$ will also ensure that the overall expected time to find Proof of Work is no larger than τ_u.

Remark 1 *If we were to study constraints that bound the overall time with a lower bound τ_l, we may have to consider other abstractions or a refined, joint probability space.*

Next, let us modify the probability that the actual time of obtaining Proof of Work is at least th. Similarly to before, we reason that

$$(th/T_i) - 1 < k \tag{43}$$

models that the *actual* time taken by physical group i to hash $k+1$ rounds is larger than th. This yields

$$prob^{s_i}(PoWTime > th) = y_i^{\lceil (th/T_i)-1 \rceil + 1} - y_i^{\lambda+1} \tag{44}$$

We want to make sure that this probability is bounded by δ from above. Intuitively, it seems ok if some physical group violates this bound, as long as at least one of the physical groups satisfies that bound. This is keeping in mind that all physical groups participate in the same mining race over a shared nonce space. So this suggests that we have the constraints:

$$\begin{aligned} & \lceil (th/T_i) - 1 \rceil \leq \lambda \qquad (\forall 1 \leq i \leq g) && (45)\\ & \delta \geq \min_{i=1}^{g} prob^{s_i}(PoWTime > th) && (46) \end{aligned}$$

Next, recall that the expression $prob^s(PowTime < th')$ denoted the probability that the *actual* time to mine would be less than th' in the original model. We can take a similar approach to this here and thus derive the definition

$$prob^{s_i}(PoWTime < th') = 1 - y_i^{\lfloor (th'/T_i)-1 \rfloor + 1} \tag{47}$$

This has the same hard constraint for the index of the summation as in (43). We get thus the constraint that

$$\delta_1 \leq \max_{i=1}^{g} prob^{s_i}(PoWTime < th') \tag{48}$$

We use operator max here instead of operator min, since we want to make sure that, *for all physical groups*, these probabilities are below δ_1.

Remark 2 *As for some of the other measures, the constraints in (46) and (48) do not explicitly model a* joint *probability distribution across all physical groups, but they seem to be a conservative approximation of the behavior such a joint distribution.*

Next, let us consider the probability of disputes. Let us derive the probability that a particular miner in physical group i finds Proof of Work within μ seconds. The derivation of this probability is similar to how this was done for the original model. For

$$w_i = (1 - 2^{-d})^{\lfloor \mu/T_i \rfloor + 1} \tag{49}$$

we get that this probability equals

$$\sum_{k=0}^{\lfloor \mu/T_i \rfloor} (1 - 2^{-d})^k \cdot 2^{-d} = 1 - w_i \tag{50}$$

The probability that no miner from physical group i finds Proof of Work is therefore

$$prob^{s_i}(0\ PoW\ within\ \mu) = (1 - (1 - w_i))^{s_i} = w_i^{s_i} \tag{51}$$

The probability that exactly one miner from physical group i finds Proof of Work is then

$$prob^{s_i}(1\ PoW\ within\ \mu) = s_i \cdot w_i^{s_i - 1} \cdot (1 - w_i) \tag{52}$$

We can use this probability to compute the probability that there is at most 1 miner *across all physical groups* who finds Proof of Work within μ seconds. Note that there are the following *disjoint* scenarios that make up this event:

- e_{zero}, as the event in which for all physical groups i, no miner from physical group i finds Proof of Work within μ seconds; so this is the event in which no miner at all finds Proof of Work within μ seconds
- e_i, as the event in which exactly 1 miner from physical group i finds Proof of Work within μ seconds, and no miner from any physical group $j \neq i$ finds Proof of Work within μ seconds.

Let us first compute the probability of event e_{zero}. Using the ROM model for hashing, we conclude that the probability that 0 miners from physical group i find Proof of Work within μ seconds is independent from that probability for any physical group $j \neq i$. Therefore, we get that

$$prob^s(0\ PoW\ within\ \mu) = \prod_{i=1}^{g} w_i^{s_i} \tag{53}$$

Similarly, we use the ROM model of hashing to infer that the probabilities for Proof of Work outcomes are independent across physical groups. Therefore, we get

$$prob^s(e_i) = s_i \cdot w_i^{s_i - 1} \cdot (1 - w_i) \cdot \prod_{j=1, j \neq i}^{g} w_j^{s_j} \tag{54}$$

Now we merely have to add up the probabilities of these $1 + g$ events e_{zero}, and e_i for $1 \leq i \leq g$ to compute

$$prob^s(no\ dispute\ within\ \mu) = \Big[\prod_{i=1}^{g} w_i^{s_i}\Big] + \sum_{i=1}^{g} s_i \cdot w_i^{s_i - 1} \cdot (1 - w_i) \cdot \prod_{j=1, j \neq i}^{g} w_j^{s_j} \tag{55}$$

From this, we only need to take the complementary probability to get

$$prob^s(dispute\ within\ \mu) = 1 - \Big(\Big[\prod_{i=1}^{g} w_i^{s_i}\Big] + \sum_{i=1}^{g} s_i \cdot w_i^{s_i - 1} \cdot (1 - w_i) \cdot \prod_{j=1, j \neq i}^{g} w_j^{s_j}\Big) \tag{56}$$

The constraints we thus want to enforce are (56) and the inequality

$$\delta_2 \geq prob^s(dispute\ within\ \mu) \tag{57}$$

Let us now summarize these findings. All definitions and constraints for the above optimization problem are shown in Figure 14. Let us formally define notation for the resulting set of constraints next.

Definition 2 *We write $\mathcal{C}^{\vec{v}}(s, r, d)$ for the set of constraints in Figure 14, where $\vec{v}$ denotes all constants and variables in that set of constraints other than s, r, and d.*

7.5 Robust Design Security for Group-Based Model

Next, we should reconsider the settings for robust design security. First, let us look at the inequality

$$|\, d - d' \,| \leq u_d \tag{70}$$

from our original model. This is a global robustness aspect and so it should be unchanged in the group-based model. We may also want the ability to bound how many miners may be corrupted in each physical group i. We can stipulate that the upper

$$
\begin{aligned}
s_l &\leq s \leq s_u \qquad r_l \leq r \leq r_u \qquad d_l \leq d \leq d_u && (58)\\
s_i^l &\leq s_i \leq s_i^u \qquad (\forall 1 \leq i \leq g) && (59)\\
s &= \sum_{i=1}^{g} s_i \qquad s_l \geq \sum_{i=1}^{g} s_i^l \qquad s_u \leq \sum_{i=1}^{g} s_i^u && (60)\\
\lambda &= \lfloor 2^r / s \rfloor && (61)\\
y_i &= (1 - 2^{-d})^{s_i} \qquad (\forall 1 \leq i \leq g) && (62)\\
\epsilon &\geq \min\{y_i^{\lambda+1} \mid 1 \leq i \leq g\} && (63)\\
\tau_u &\geq \min\{T_i \cdot \frac{1 - y_i^{\lambda+1} - (\lambda+1) \cdot (1 - y_i) \cdot y_i^{\lambda+1}}{1 - y_i} \mid 1 \leq i \leq g\} && (64)\\
\lambda &\geq \lceil (th/T_i) - 1 \rceil \qquad (\forall 1 \leq i \leq g) && (65)\\
\delta &\geq \min\{y_i^{\lceil (th/T_i)-1 \rceil + 1} - y_i^{\lambda+1} \mid 1 \leq i \leq g\} && (66)\\
\delta_1 &\leq \max\{1 - y_i^{\lfloor (th'/T_i)-1 \rfloor + 1} \mid 1 \leq i \leq g\} && (67)\\
w_i &= (1 - 2^{-d})^{\lfloor \mu / T_i \rfloor + 1} \qquad (\forall 1 \leq i \leq g) && (68)\\
\delta_2 &\geq 1 - \Big(\Big[\prod_{i=1}^{g} w_i^{s_i} \Big] + \sum_{i=1}^{g} s_i \cdot w_i^{s_i - 1} \cdot (1 - w_i) \cdot \prod_{j=1, j \neq i}^{g} w_j^{s_j} \Big) && (69)
\end{aligned}
$$

Figure 14: Set of constraints $\mathcal{C}^{\vec{v}}(s, r, d)$ (see Definition 2) for our optimization problem of group-based Proof of Work, *without* modeling strict uncertainty for robust optimization and *without* representing consortium partners and constraining their hash power. There are $g > 1$ physical groups of miners with s_i mining units at hash rates T_i within physical group i. Integer variables are s, r, d, s_i and integer constants are s_l, s_u, r_l, r_u, d_l, d_u, s_i^l, s_i^u, l_i, and c. The two inequalities in (60) are not required but they are sanity checks on that the bounds s_u and s_l really say something useful. There are real-valued, non-negative constants as well: ϵ, τ_u, T_i, th, μ, th', δ, δ_1, δ_2, and δ_3 where the latter four denote probabilities in $[0, 1]$. Variables λ, y_i, w_i, and integer variable s are auxiliary ones in that there value is determined by constants and values of other variables

bound on the number of overall corrupted miners is not larger than the sum of all such upper bounds for corrupted miners from each physical group:

$$u_s \leq \sum_{i=1}^{g} u_i^s \tag{71}$$

It makes sense to constrain both u_s and u_i^s and making the above not an equality allows us to model, for example that in each physical group up to 4 miners may be corrupted whereas in total up to 6 miners may be corrupted.

As additional constraints for *strict uncertainty* we consider

$$s_i - u_i^s \leq s_i' \leq s_i \qquad (\forall 1 \leq i \leq g) \tag{72}$$

$$|\, d - d' \,| \leq u_d \tag{73}$$

$$s - u_s \leq \sum_{i=1}^{g} s_i' \leq s \tag{74}$$

where s_i' refers to the actual number of miners in physical group i under strict uncertainty, and d' is the actual level of difficulty (shared across all physical groups) under strict uncertainty.

We can now define the robust cost function, where we may identify s' with either the sum of all s_i' or the tuple of all such values, as appropriate:

$$robustCost(s, r, d) = \max\{Cost(s', r, d') \mid \text{constraints in (72)-(74) hold}\} \tag{75}$$

where $Cost(s', r, d')$ is defined as in (31). Note that this robust cost function also implicitly depends on the above constants such as u_s and u_i^s.

We can now define the predicate that evaluates whether a tuple (s, r, d) is robustly feasible in this group-based setting:

$$robustFeasible(s, r, d) = \bigwedge_{(s', r, d') \mid (72)-(74) \text{ holds}} \mathcal{C}^{\vec{v}}(s', r, d') \tag{76}$$

As in the original model, function $robustFeasible(s, r, d)$ is true iff the *non-robust version* of the constraint set from Figure 14 is feasible for all triples (s', r, d') that satisfy constraints (72)-(74).

We now describe a robust optimization problem shown in Figure 15. This considers two physical types of miners where up to 4 miners from each physical type may be corrupted or fail but at most 6 miners in total may have such behavior. The uncertainty in the level of difficulty is 3 and the probability that 6 consecutive blocks are won within any physical group by 6 of its miners is bounded by $\delta_3 = 0.001$.

7.6 Modeling a Consortium and its Partners

Let us now consider a consortium of $P > 1$ partners. We extend the above mathematical model with a mapping of machines to partners. For each physical type i, we identity the

$\min\{robustCost(s,r,d) \mid robustFeasible(s,r,d)\}$
subject to the set of constraints from Figure 14 together with

$g = 2$	$u_1^s = u_2^s = 4$	$u_s = 6$	$u_d = 3$
$c = 6$	$l_1 = l_2 = 6$	$0.001 = \delta_3$	$l^c \leq s^c \cdot \delta_3$

Figure 15: Robust optimization problem for two types of miners where up to 4 miners from each type may be corrupted or fail but at most 6 miners in total may have such behavior. The uncertainty in the level of difficulty is 3 and the probability that 6 consecutive blocks are won within any group by 6 of its miners is bounded by $\delta_3 = 0.001$

s_i machines of that type with the integers $\{1, 2, \ldots, s_i\}$. We also identity consortium partners with integers $\{1, \ldots, P\}$. The function

$$owner_i \colon \{1, \ldots, s_i\} \to \{1, \ldots, P\} \tag{77}$$

then specifies which parter owns/controls which machine of type i: so that $owner_i(j) = l$ means that machine j of type i is controlled by partner l. For each parter l and physical type i, let l_l^i be the number of machines of type i that are controlled by l. Then $\mathsf{l}_l^i \cdot T_i^{-1}$ is the hash rate that partner l has over all machines of type i that she controls. Therefore,

$$hr_l = \sum_{i=1}^{g} \mathsf{l}_l^i \cdot T_i^{-1} \tag{78}$$

is the total hash rate of partner l across all machines (of all physical types) that she controls. We can therefore express the *relative hash power* of a partner l within the consortium by the expression

$$hashPower(l) = \frac{hr_l}{\sum_{l' \neq l} hr_{l'}} \tag{79}$$

We can use this measure to express stability requirements for the consortium hash power, which we also want to verify at run-time. For example, we wish to bound each partners relative hash power as in

$$\max(0, low(l)) < hashPower(l) < \min(0.33, high(l)) \qquad (\forall 1 \leq l \leq P) \tag{80}$$

where the 0.33 is making sure that mining attacks based on 33% of the mining pool cannot be launched. The functions $low(l)$ and $high(l)$ may be arbitrarily complex and informed by data analytics and other inputs.

A concrete case worth exploring in the next paper project may be $g = 3$ and $P = 3$. Perhaps one could also get some experimental test data for this. We also want to think about the engines (first and second order ones) that could approximate the above model and its optimization in near real-time. As engine input we may want, for each block from the chain within some finite horizon:

- the identify and physical type of the miner who won that block
- the values of r and d for which the block were won
- the time it took to mine this since the last block
- possibly the values of g and P at that time
- the partner who owns the miner who won this last block

The output of this engine which at least contain r and d, so that it can learn how best to adapt the system to optimize for its resiliency. This is akin to statistical machine learning and the use of information theory, to match the real data as close as possible to the mathematical model.

8 Machine Learning Run-Time Stabilization

8.1 Machine Learning Engines in Permissioned Blockchains

Given a budget, an engine optimizes for stability by adjusting internal parameters of a Permissioned Blockchain at run-time. Optimization decisions taken by an engine are based on a continually evolving machine learning model.

Model We define the set of all possible states the system can be in as B (i.e. the last k blocks from the chain including operational metadata concerning those blocks). The set of all possible run-time parameters an engine can control is defined as C. With that an engine can be defined as the function

$$\mathit{engine}\colon B \to C$$

mapping from blockchain state to control parameter. Three steps determine the mapping:

- Feature preprocessing transforming B to neural network parameters S
- Neural network approximation of optimal policy function $\pi\colon S \to A$
- Action validation and mapping to control parameters: $\mathit{validate}\colon A \to C$

Overall, an *engine* can be formally described as a composite function:

$$\mathit{engine}(b) = (\mathit{validate} \circ \pi \circ \mathit{preprocess})(b)$$

8.2 Reinforcement Learning for Adaptive Engines

The crucial element of our engine is the machine learning model responsible for taking decisions on how to adapt run-time parameters. While we discuss the model representation in a later section, we want begin by discussing possible training methods.

One approach would be to use supervised learning: Using a large amount of labeled data, we would train a model that can reproduce parameter adjustment suggested by the training data. This approach however has several drawbacks:

- Non-trivial data collection: Gathering the required amount of training data is non-trivial and expensive. A human operator would have to monitor a running blockchain and take (best guess) decisions that get recorded.
- Inherent bias: The resulting data is tightly coupled to the system the blockchain was running on. A blockchain running in a datacenter will react very differently to parameter changes than a blockchain running on low-powered IoT clients (due to vastly different system characteristics like miner spin-up latency or network latency).
- Static nature: The model will only be able to take decisions implied by the training data and it is unlikely to surpass the performance of those decisions.

To overcome those drawbacks we suggest to use reinforcement learning to train machine learning engines. One can think of reinforcement learning (RL) to be positioned in between supervised learning and unsupervised learning: Whereas supervised learning has labels for each training example and unsupervised learning has no labels at all, reinforcement learning has sparse and time-delayed labels.

The unique properties of RL enable an agent to build a model of an environment solely by observing its state transitions. As an agent interacts with an environment, it builds an internal representation of how it believes the environment to be working and adapts its own behavior to be successful in this environment. Two types of adaptiveness are worth highlighting:

- Systemic adaptiveness: Any system that a blockchain can run on will exhibit different characteristics (e.g. miner spin-up latency). An ideal model will adapt to those characteristics as it is impossible to express the full set of performance-influencing factors as model parameters.
- Temporal adaptiveness: Some intrinsic characteristics of a system can change over time (e.g. decreased network latency due to hardware upgrades). Such changes are again not fully representable as model parameters so an ideal model would adapt to changing environments as time progresses.

Reinforcement Learning Introduction We'll use the example of self-driving cars to introduce the basic ideas used in reinforcement learning. As briefly mentioned, there are two entities interacting with each other: Agent and Environment (Figure 16).

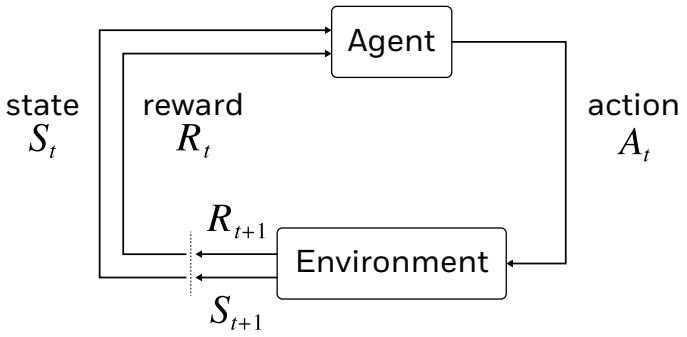

Figure 16: Agent Environment interaction [30]

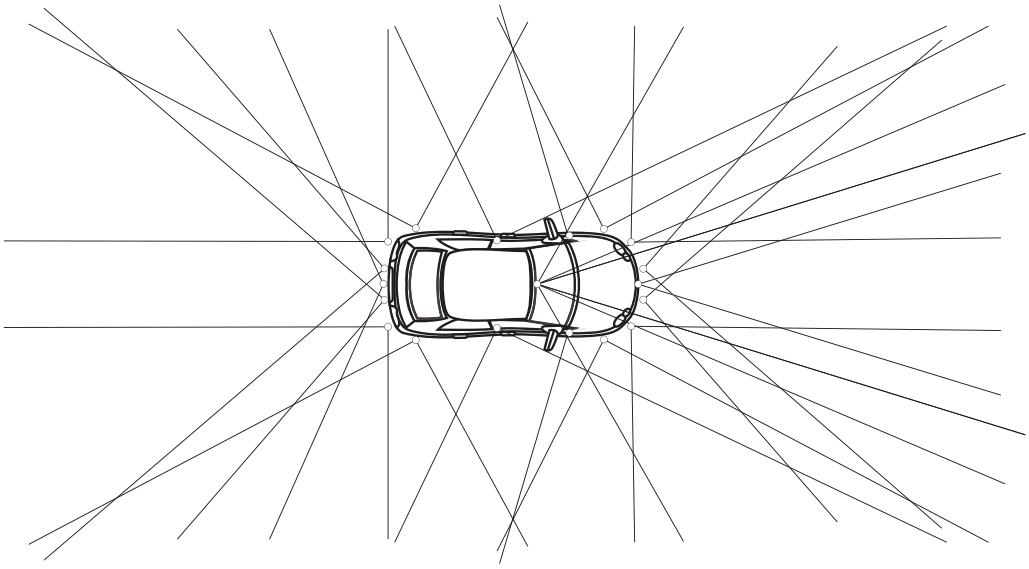

Figure 17: Autonomous car perceiving state of its environment

In the case of autonomous driving, the agent would be an autonomous car, and the environment would be an actual environment the car is placed in: streets, traffic signs, other cars, pedestrians, and other factors. The agent has a set of actions it can carry out in the environment, e.g. accelerate, decelerate, turn left, turn right. It does so after observing the state of the environment. An autonomous cars' observation of the environment (the state the environment is in) is the data it receives from e.g. cameras, GPS, and ultrasonic sensors (see Figure 17).

Terminal states The agent carries out actions in the environment and observes state changes. States can be terminal: Once the agent caused the environment to be in a terminal state, the simulation episode is over. This can be either because the agent was successful in accomplishing a task (self-driving car arrived at destination), or because the agent caused an undesired outcome (self-driving car accident).

Rewards State changes are sometimes (but not always) associated with feedback from

the environment (positive or negative). Those sparse and time-delayed labels in RL are called rewards: A self-driving car will get rewarded when it arrives safely at the destination.

The sparse and time-delayed nature of rewards leads to the **credit assignment problem**: Which action (and to which extend) was responsible for receiving a reward? Usually it was not the last action just before the reward, but a combination of previous actions: Safely overtaking the bicycle, stopping at the red light, taking the correct turns.

Defining (and assigning) the right rewards is essential for reinforcement learning: To avoid undesired terminal states, one usually defines a robust target to optimize for. In the case of overtaking a cyclist, contact between car and cyclist can be considered a terminal state. One would set e.g. a 1.5m distance as optimal robust target, with decreasing (and even negative) rewards the further the agent diverges from this target: We don't want the car to overtake a cyclist with just a few centimeters of distance as it might scare the cyclist and not account for surprising movements, so a negative reward is assigned. We also don't want the distance to be too big: The maneuver might surprise other drivers.

8.3 Environment Agent Model for Permissioned Blockchains

State We represent state S using k triplets each holding the mining time, difficulty, and approximated hash rate (at the time of mining block k) of the top k blocks of the chain.

$$S = (hash\_rate, mining\_time, difficulty)^k$$

Reward Agent behavior follows rewards associated with (s, a) pairs. Following our optimization objective (i.e. a maximum mining time of $target$ seconds), we define

$$robust\_target = 0.8 * target$$

to determine rewards. We assign positive rewards if the agent is close to the $robust\_target$, and decreasing (even negative) rewards the further an agent diverges from $robust\_target$:

$$reward = -m * (robust\_target - mining\_time_k)^2 + 1$$

The scale of the target can be adjusted using parameter m.

Terminal state The function $terminal\colon S \to \{True, False\}$ returns $True$ if state $s \in S$ is a terminal state:

$$terminal(S) \begin{cases} \text{True}, & \text{if } mining\_time_k < robust\_target - threshold \\ \text{True}, & \text{if } mining\_time_k > robust\_target + threshold \\ \text{False}, & \text{otherwise} \end{cases}$$

Action Space Run-time parameter adjustments are expressed as a set of actions A, affecting both the level of difficulty d and the number of miners currently mining:

$$A = \{d\_increase, d\_decrease, miner\_start, miner\_stop\}$$

Scalability The model presents a minimalistic approach concerning the proposed elements in A and S. It has been chosen to represent a framework that scales well to more complex scenarios: Additional run-time adjustable parameters can be added to A, additional features can be added to S, and reward assignment can be adjusted by modifying r.

8.4 Q-learning

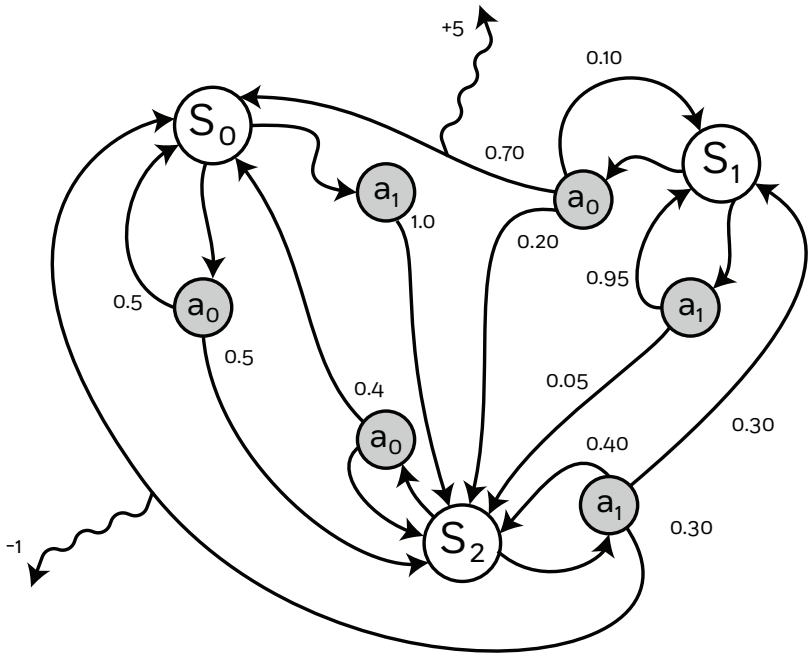

Figure 18: Markov Decision Process with transition probabilities and rewards

Formalism Reinforcement learning problems can be formalized as Markov Decision Processes. A Markov Decision Process consists of a set of states S, a set of actions A, rules for state transitions, and rules for rewards (Figure 18). One run through the Markov process (from initial state to terminal state) is represented by a finite sequence of states, actions and rewards. Individual runs are often called episodes:

$$s_0, a_0, r_1, s_1, .., s_{n-1}, a_{n-1}, r_n, s_n \qquad (\text{with } terminal(s_n) = True)$$

Discounted Future Reward Looking at one episode and starting at time t, the total reward from t to the end of the episode is defined as

$$R_t = r_t + R_{t+1}$$

Since our Markov Decision Process is stochastic, the further we look into the future the less confident we can be about upcoming rewards. The concept of Discounted Future Reward accounts for that by using parameter γ (with $0 \leq \gamma \leq 1$) to discount expected rewards.

$$R_t = r_t + \gamma R_{t+1}$$

An agent would then choose a strategy that maximizes R, i.e. at any time t choose action a that provides the highest discounted future reward. Parameter γ will play a crucial role in the behaviour of our algorithm: Small values of γ favour immediate rewards while large values encourage long-term optimization. But how would an agent observing state s know which action a to take?

Function $Q(s, a)$ represents the discounted future reward when an agent performs action a in state s and continues to choose the best possible action in each state for the remainder of the episode. Assuming the existence of such a function $Q(s, a)$, an agent can easily decide which action to take in each state: The one that maximizes the discounted future reward. This is formalized in the policy function π which (assuming the existence of $Q(s, a)$) decides which action to take in which state:

$$\pi(s) = argmax_a Q(s, a)$$

Bellmann equation The Bellman equation defines the Q-function assuming that we already know the discounted future reward for each $(state, action)$ pair:

$$Q(s, a) = r + \gamma max_{a'} Q(s', a')$$

The discounted future reward for taking action a in state s is defined as the sum of immediate reward r and maximum discounted future reward of the following state s'. Q-learning uses the Bellman equation to iteratively approximate the ideal Q-function Q^*. Although initial future rewards are random, updating those values with actually experienced rewards has been proven to converge [26]. The details of how the iterative approximation is carried out in practice highly depends on the way we choose to represent our Q-function.

8.5 Adaptive Neural Network Engines

How can the Q-function be represented? A simple representation would be to have a Q-table, with one row for each possible state the environment can be in ($s \in S$), and one column for each possible action ($a \in A$). However this approach does not scale well with the number of features: For some inputs the number of features grows rather quickly (e.g. a single 1024x768 photo would require $256^{3 \times 1024 \times 768}$ features for three RGB layers), resulting is a large number of rows in the table. Even if the number of features is still tangible, many of those possible states would rarely be encountered, resulting in a sparse Q-table that is unlikely to converge in a lifetime.

Neural networks on the other hand are known to be good at capturing complex hypothesis. The influential DeepMind paper "Playing Atari with Deep Reinforcement Learning" [28] proposes a network architecture that takes state as input and outputs Q-values for every possible action. Our neural network approximating the Q-function would therefore have $3 * k$ input units (each input unit representing one attribute of the top k blocks), a number of hidden layers, and $|A|$ output units.

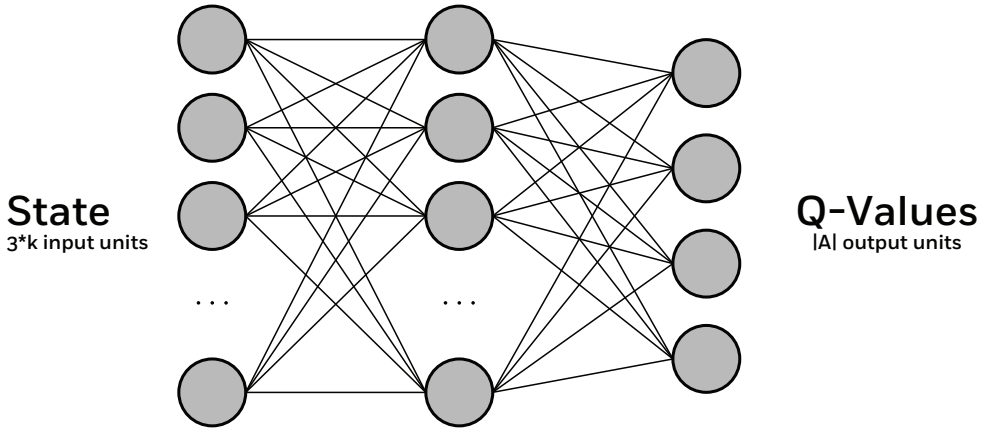

Figure 19: Neural Network Architecture for Reinforcement Learning

The benefit of this architecture is that it enables a single forward propagation pass to give us Q-values for all possible actions at once.

8.6 Training Adaptive Neural Network Engines

Exploration-Exploitation Dilemma Initially, Q-values are random, so an agent can only guess which action might yield a result (positive reward). Over time, as rewards are experienced, an agent might use this experience to chose actions that promise increased rewards. The question is: How should an agent decide when to pick actions at random and when to take actions that promise the highest reward?

A solution to this is called **ϵ-greedy exploration**. Parameter ϵ (with $0 < \epsilon < 1$) controls the exploration/exploitation tradeoff. Random actions get picked with probability ϵ, otherwise the path that promises the highest future reward is chosen. In practice, it is common to start with a high value for ϵ (high probability for random exploration) and decrease the value over time as the approximation of Q gets more and more accurate.

Training To find the weights of our neural network ("train" the network), we need to define an error function that compares predicted rewards to actually experienced rewards ("target"). In this sense, training a neural network using reinforcement learning is not too different from training a neural network using supervised learning. We chose

to use a standard squared error loss function

$$loss = (reward + \gamma \max_{a'} Q(s', a') - Q(s, a))^2$$

where $reward + \gamma \max_{a'} Q(s', a')$ is the target and $Q(s, a)$ is the current prediction of our Q-network. Again, parameter γ controls the discounting of future rewards.

By interacting with the environment using actions, an agent observes state transitions (along with occasional rewards). Those observations can be "bundled" in one object called an experience: $< s, a, r, s' >$. To update our neural network using those experiences, we do the following:

1. Forward-propagation pass using s as input to get the current $predictions_s$ Q-value vector
2. Forward-propagation pass using s' as input to get $Q(s', a')$ predictions (for all $a' \in A$)
3. Calculate the target for a as $target_a = r + \gamma \max_{a'} Q(s', a')$ (which is the sum of immediate reward r and maximum discounted future reward)
4. Define vector $target$ such that:
   $target[a] = target_a \wedge \forall a' \in A \setminus \{a\} \bullet target[a'] = predictions[a']$
5. Use $target$ with backpropagation to update θ

Experience Replay With reinforcement learning, subsequent experiences are often similar in nature. Usually just a few dimensions of the state change while other dimensions remain unchanged, e.g. an autonomous vehicle advances its position but the street, surrounding vehicles, weather condition and speed limit remain the same (or at least very similar). This poses the challenge since training neural networks with many similar subsequent samples can result in undesired outcomes, like converging to a local minima.

To avoid this, a technique called **experience replay** can be used. Experiences observed by an agent get stored in a cache called replay memory. For training, random samples are drawn from this replay memory, thereby avoiding subsequent training examples that are too similar. The full Q-network training algorithm (including ϵ-greedy exploration and experience replay) can be seen as pseudocode in Figure 8.6.

8.7 Prior Work

C.J.C.H. Watkins introduced Q-learning in his Ph.D. thesis at Cambridge University in the year 1989 [25]. It was followed by a convergence proof (Watkins and Dayan) in 1992 [26]. A more recent milestone in the application of Q-learning was DeepMind achieving expert human levels at playing a selection of Atari 2600 games which resulted in a patent grant for Google [27].

```
begin
    initialize replay memory D;
    initialize Q with random weights;
    ε = 1.0;
    while iteration < num_iterations do
        s = environment : observe initial state;
        while not terminal(s) do
            a = max_{a'} Q(s, a');
            with probability ε : a = random action ∈ A;
            s', reward = environment : carry out a;
            store < s, a, r, s' > in replay memory D;

            sample random experiences from replay memory D;
            train network using sample;

            s = s';
        end while
        decrease ε;
        iteration += 1
    end while
end
```

Figure 20: Reinforcement learning with experience replay and ϵ-greedy exploitation (adapted from [29])

Our work differs from Google's Deep Q-Networks in the substantial way that their network processes video data frames using multiple convolutional layers, whereas our network is optimized for time-series data of blockchain mining behavior.

8.8 Action Validation

Before an action proposed by the engine (or more precisely, the neural network) gets propagated through the mining network, a final validation layer validates the action and translates it into a valid control command. Receiving the action *d_increase* the validation might e.g. check if d would cross some upper bound d_{upper} after being increased. The validation layer exists to incorporate static business logic in the engine. Another example would be the business decision to have at least three miners running at any point in time. If the current machine count equals three, and the engine decides (e.g. due to ϵ-greedy exploration) to issue a *miner_stop* action, the validation layer can prevent this from happening, resulting in a no-op.

During training, the *engine* will experience that certain sequences of actions have no effect and adjust its behavior (model) accordingly. Coming back to the autonomous

vehicle examaple, the autonomous driving AI will learn that hitting the brakes in a standing car does not cause any change in the state of the environment.

8.9 Feature Adaptions for Governed Blockchain Engines

The chosen framework of using reinforcement learning with a Q-function neural network approximation scales well and enables us to integrate additional features without much effort. In the context of governed blockchains additional features are required to express the mining balance between different partners participating in the mining process. Designing those features deserves some attention: Informally speaking, we want features that do not vary too much when e.g. the number of partners P changes, but that have a strong signal when balance among those partners changes.

As an example, from input "partner who owns the miner who won the block" we can model an aggregated feature $\sigma_{winning\_partner}$ that expresses how "balanced" the winning of mining races is across partners (standard deviation):

$$\sigma_{winning\_partner} = \sqrt{\frac{\sum_{i=1}^{P}(w_i - \overline{w})^2}{N-1}}$$

with w_i being the number of blocks won by machines owned by partner i.

9 Experiments and Validation

We submitted simple instances of the optimization problem in Figure 13 to state of the art MINLP solvers. All these solvers reported, erroneously, in their preprocessing stage that the problem is infeasible. These solvers were not designed to deal with problems that combine such small numbers and large powers, and rely on standard floating point implementations. Therefore, we wrote a bespoke solver in Haskell that exploits the fact that we have only few integral variables within limited ranges so that we can explore their combinatorial space completely to determine feasibility and therefore optimality as well.

Experimental Setup We solve the robust optimization problem for the analytical expressions we derived above with the algorithm depicted in Figure 9. This algorithm has as input the set of constraints, a parameter p and a parameter α. It will output at most p robustly feasible tuples $(s, r, d, cost)$ from a list of all robustly feasible such tuples as follows: it will identify the maximal values of d for which such tuples are robustly feasible, and it will report exactly one such tuple for each value of d where r is minimal, and *cost* is minimal whilst also bounded above by $\alpha \cdot c_m$ where c_m is the globally minimal cost. This also determines the values of s in these tuples and so the algorithm terminates.

Now, having defined the required analytical expressions and the algorithm to report the best p robustly feasible tuples in Figure 9, we also want to validate these expressions

input : p, α, and values for all constants in Figure 13
invariant: *list* lists tuples $(s, r, d, cost)$ in descending order for d

begin
  define all constants for constraints in Figure 13;
  $list = [(s, r, d, cost) \mid cost = Cost(s, r, d), feasibleFloat(s, r, d)\ is\ true]$;
  $list = [(s, r, d, cost) \in list \mid feasibleFloat^{u_s}_{u_d}(s, r, d)\ is\ true]$;
  while $(\exists (s, r, d, cost) \neq (s', r', d, cost') \in list)$ **do**
    case $(cost' < cost) \vee (r' < r)$ **do** *remove* $(s, r, d, cost)$ *from list*);
    case $(cost < cost') \vee (r < r')$ **do** *remove* $(s', r', d, cost')$ *from list*);
  end while
  $c_m = \min\{c \mid \exists (s, r, d, cost) \in list\}$;
  while $(\exists (s, r, d, cost) \in list \colon cost > \alpha \cdot c_m)$ **do**
    remove $(s, r, d, cost)$ *from list*;
  end while
  $results = list\ of\ first\ p\ tuples\ from\ list$;
  $results = [(s, r, d, cost) \in results \mid feasibleBigFloat^{u_s}_{u_d}(s, r, d)\ is\ true]$;
  return *results;*
end

Figure 21: Algorithm, written in imperative style of list processing, for reporting the best p robustly feasible tuples $(d, r, s, cost)$ such that d is maximal subject to the cost $cost = Cost(s, r, d)$ satisfying $cost \leq \alpha \cdot c_m$ where c_m is the minimal cost for all robustly feasible tuples (s, r, d) and $\alpha \geq 1$ is a tolerance factor for increasing cost beyond c_m. Predicate $feasibleFloat(s, r, d)$ is true iff all constraints in Figure 12 are true for this choice of s, r, and d under normal precision floats. Predicates $feasibleBigFloat$ and $feasibleBigFloat^{u_s}_{u_d}$ are true iff their mathematical definition is true under arbitrary-precision floating points (applying package `Data.BigFloat` version 2.13.2).

and the algorithm experimentally. Our setup for this is based on pure Haskell code, as functional – and in particular – Haskell programs offer the advantages of being modular in the dimension of functionality, being strongly typed as well as supporting an easy deconstruction of data structures, particularly lists [3]. Furthermore, the arbitrary-precision verification is handled by the external `Data.BigFloat` package, which is also written in Haskell. Further verification and validation of the received results are pursued by unit testing using an arbitrary precision calculator. Moreover, our experiments ran on a machine with the following specifications: Intel(R) Xeon(R) CPU E5-4650 with 64 cores and 2.70GHz and 500 GB total RAM. Our machines required between 322.12 and 261.425 seconds to compute the respective optimizations. The entire experiment took 10,457.58 seconds.

We instantiate the model in Figure 13 with the constants shown in Table 1. We choose T to be $1/(50 \cdot 10^9) = 0.02 \cdot 10^{-9}$ for a mining ASIC from early 2016 with an

$s_l = 4$	$s_u = 80$	$r_l = 24$	$r_u = 64$
$d_l = 4$	$d_u = 64$	$\text{TVC} = 2 \cdot 10^{-12}$	$\text{TFC} = 3000$
$\alpha = 1.5$	$T = 0.002 \cdot 10^{-9}$	$th = 300$	$th' = 300$
$\delta = 10^{-9}$	$\delta_1 = 1$	$\delta_2 = 0.001$	$\delta_3 = 0.001$
$\tau_l = 0$	$\mu = 1/10000$	$\epsilon = 2^{-64}$	$k = 5$
$u_d = 3$	$u_s = 5$	$c = 6$	$l = 4$

Table 1: Constants for our experiments. This does not specify the values of τ_u which will vary in experiments. Some experiments will also vary the values of δ, δ_2 or δ_3

estimated cost of 2700 USD at that time, so a fixed cost of TFC = 3000 USD seems reasonable. Let us now explain the value $2 \cdot 10^{-12}$, which models the energy cost of a sole hash (we can ignore other costs on that time scale). A conservative estimate for the power consumption of an ASIC is 10 watts per Gigahashes per second, i.e. 10 watts per Gh/s. We estimate the cost of one kilowatt hour kWh to be about 10 cents. A kWh is $3600s$ times kW and one kW is 1000 watts. So 10 watts per Gh/s equals $10 \cdot 3600$ watts, which amounts to $36kWh$. So the cost for this is $36 \cdot 10$ cents per hour, i.e. 360 cents per hour. But then this costs $360/3600 = 0.1$ cents per second. The price for a sole hash is therefore 0.1 divided by $50 \cdot 10^9$, which equals $\text{TVC} = 2 \cdot 10^{-12}$.

We insist on having at least 4 miners and cap this at 80 miners. The *shared* nonce space for miners is assumed to be between 24 and 64 bits. The level of difficulty is constrained to be between 4 and 64. We list optimal tuples that are within a factor of $\alpha = 1.5$ of the optimal cost. We make the value th' irrelevant by setting $\delta_1 = 1$ which makes the constraint for th' vacuously true. The probability for mining failure is not allowed to exceed $\epsilon = 2^{-64}$. Setting $\tau_l = 0$ means that we don't insist on the average mining time to be above any particular positive time. The probability that mining a block takes more than $th = 300$ seconds is bounded by 10^{-9}. And the probability that more than one miner finds PoW within $\mu = 1/10000$ seconds is bounded by 0.001, which we also use as bound for winning 6 consecutive mining races. The algorithm reports the top $k = 5$ optimal tuples – and reports fewer if there are no 5 feasible tuples. The remaining constants for robustness are as given in Figure 13.

Let us now specify some values of τ_u of interest. As reported in [6], Bitcoin is believed to handle up to 7 transactions per second (although this can be improved [8]), Paypal at least 100 transactions per second (which we take as an average here), and Visa anywhere between 2000 and 7000 transactions per second on average. By *transactions per second* we mean that blocks are mined within a period of time consistent with this. Of course, this depends on how many transactions are included in a block. For sake of concreteness and illustration, we take an average number of transactions in a Bitcoin block, as reported for the beginning of April 2016, that is 1454 transactions.

For a Bitcoin style rate, but in our *governed* setting, this means that a block is mined in about $1454/7 \sim 207.71$ seconds. Since $T \cdot E^s(noR)$ is the expected (average) time to mine a block, we can model that we have 7 transactions per second on average

$\tau_u^{Bitcoin}\ (s, r, d, cost)$	$\tau_u^{PayPal}\ (s, r, d, cost)$	$\tau_u^{Visa}\ (s, r, d, cost)$
$(18, 48, 41, 54004.4)$	$(18, 48, 41, 54004.4)$	$(18, 48, 35, 54000.07)$
$(18, 48, 40, 54002.2)$	$(18, 48, 40, 54002.2)$	$(18, 48, 34, 54000.035)$
$(18, 48, 39, 54001.1)$	$(18, 48, 39, 54001.1)$	
$(18, 48, 38, 54000.55)$	$(18, 48, 38, 54000.55)$	
$(18, 48, 37, 54000.27)$	$(18, 48, 37, 54000.27)$	

Table 2: Output for top 5 optimal tuples for our robust optimization problem run in configuration C1 and with values τ_u as listed in (81): 5 optimal tuples are found for $\tau_u^{Bitcoin}$ and τ_u^{PayPal}, i.e. at least 5 values of d are feasible. The problem has two feasible levels of difficulty for τ_u^{Visa}. Costs are rounded up for three decimal places

by setting $\tau_u^{Bitcoin}$ to be $1454/7$. Similarly, we may compute τ_u^{PayPal} and τ_u^{Visa} based on respective 100 and 7000 transactions per second:

$$\tau_u^{Bitcoin} = 1454/7 \qquad \tau_u^{PayPal} = 1454/100 \qquad \tau_u^{Visa} = 1454/7000 \tag{81}$$

Experimental Results We now discuss the results of our experiments. Each experiment is conducted in three different configurations:

C1 constants in as Table 1, i.e. $\delta = 10^{-9}$, $\delta_2 = \delta_3 = 0.001$

C2 smaller δ, that is $\delta = 2^{-64}$, $\delta_2 = \delta_3 = 0.001$

C3 smaller δ and δ_3, that is $\delta = 2^{-64}$, $\delta_2 = 0.001$, and $\delta_3 = 0.0001$.

Transactions per second as in Bitcoin, PayPal, and Visa. We show in Table 2 output for the top 5 optimal robustly feasible tuples for the various values of τ_u in (81) for configuration C1. We see that all three transaction rates can be realized with 18 miners and a 48-bit shared nonce space in our governed setting, and this gives each miner a nonce space of about 43 bits. The achievable level of difficulty (within the uncertainty in u_s and u_d) ranges from 37 to 41 for both the Bitcoin style rate and the PayPal style rate. For the Visa style rate, the feasible levels of difficulty are 34 and 35. For the optimal tuples reported in Table 2, the value of r remains feasible whenever $48 \leq r \leq 64$. Note that these results also imply that, for all three rate styles, feasibility requires at least 18 miners.

Let us run this experiment in configuration C2. This models that the probability of mining to take more than 300 seconds is very small. We now only report the changes to the results shown in Table 2 for the top rated, optimal tuple. For $\tau_u^{Bitcoin}$, the level of difficulty drops from 41 to 40 but there are still 18 miners and a shared nonce space of 48 bits. This tuple $(s, r, d) = (18, 48, 40)$ is also optimal for τ_u^{PayPal} now, whereas the optimal tuple $(s, r, d) = (18, 48, 35)$ for τ_u^{Visa} from configuration C1 remains to be optimal for C2.

$Bitcoin \equiv 7$ $(s, r, d, cost)$	$PayPal \equiv 100$ $(s, r, d, cost)$	$Visa \equiv 7000$ $(s, r, d, cost)$
$(18, 48, 41, 54004.4)$	$(18, 48, 41, 54004.4)$	$(18, 48, 40, 54002.2)$
$(18, 48, 40, 54002.2)$	$(18, 48, 40, 54002.2)$	$(18, 48, 39, 54001.1)$
$(18, 48, 39, 54001.1)$	$(18, 48, 39, 54001.1)$	$(18, 48, 38, 54000.55)$
$(18, 48, 38, 54000.55)$	$(18, 48, 38, 54000.55)$	$(18, 48, 37, 54000.27)$
$(18, 48, 37, 54000.27)$	$(18, 48, 37, 54000.27)$	$(18, 48, 36, 54000.138)$

Table 3: Output for top 5 optimal tuples for our robust optimization problem running in configuration C1 and with values τ_u given as 50000/7, 50000/100, and 50000/7000 (respectively). Results for the first two columns are identical with those in the first two columns of Table 2. The first 4 optimal tuples for $\tau_u = 50000/7000$ equal that last 4 of the 5 optimal tuples for 50000/7. Costs are rounded up for three decimal places

Next, we run this experiment for configuration C3, also decreasing the probability that corrupt miners can win 6 consecutive mining races. For $\tau_u^{Bitcoin}$ and for τ_u^{PayPal}, the top 5 optimal tuples are $(s, r, d) = (24, 49, d)$ where $36 \leq d \leq 40$. In particular, this requires at least one more bit for the nonce space and at least 6 more miners. For τ_u^{Visa}, only tuples $(24, 49, 35)$ and $(24, 49, 34)$ are reported, so this also requires at least 24 miners and a 49-bit nonce space, where 35 and 34 are the feasible levels of difficulty.

We may explore the *feasibility boundary* for τ_u for configuration C2. The robust optimization problem is infeasible for $\tau_u = 0.06871$ but becomes feasible when τ_u equals 0.06872. In that case, the only feasible tuples are $(s, r, d) = (18, r, 34, 54000.03)$ where $48 \leq r \leq 64$.

Larger transaction rates per second. Next, we want to vary the average number of transactions *ant* in a block from $ant = 1454$ to larger values. This is sensible for our use case as transactions only record a hash, which may be 8 bytes each. These results are seen in Table 3 for 50000 transactions on average in a block, running in the configuration C1. Let us discuss the impact of changing the *ant* in a block from 1454 to 50000. This has no impact when 7 or 100 transactions per second are desired. For 7000 transactions per second, this robust optimization problem still has the same s and r values in optimal tuples but the level of difficulty (which was 35 or 34) can now be between 36 and 40. This quantifies the security and availability benefits from packing more transactions into a block for mining throughput.

Let us now see how these results change when we run the experiment in configuration C2. Now, all three rate styles report the same optimal 5 tuples which are equal to the tuples listed in the rightmost column in Table 3: $(s, r, d) = (18, 48, d)$ where $36 \leq d \leq 40$. The results for configuration C3 are also identical for all three rate styles, they equal $(s, r, d) = (24, 49, d)$ where $36 \leq d \leq 40$. So this requires one more bit in the nonce space and at least 6 more miners.

Feasibility boundary for transaction rates per second. We repeat the last ex-

periment by varying the *ant* from 50000 to half a million, in increments of 50000. We summarize these results as follows:

- *Configuration C1:* For all three rate styles and all transaction values in increments of 50000 up to 500000, the optimal tuples are the same: $(s, r, d) = (18, 48, d)$ where $37 \leq d \leq 41$.
- *Configuration C2:* For all three rate styles and all transaction values in increments of 50000 from 100000 up to 500000, the optimal tuples are the same: $(s, r, d) = (18, 48, d)$ where $36 \leq d \leq 40$. In contrast, for $50000/x$ where x is 7, 100 or 7000, we need at least a 49-bit nonce space and at least 24 miners.
- *Configuration C3:* For all three rate styles and all transaction values in increments of 50000 up to 500000, the optimal tuples are the same: $(s, r, d) = (24, 49, d)$ where $36 \leq d \leq 40$.

Range of feasible sizes for nonce space. We can compute and validate whether a robustly feasible tuple $(s, r, d, cost)$ has any other values r' for which $(s, r', d, cost)$ is robustly feasible. For example, for all the optimal tuples $(s, r, d, cost)$ we computed above, we conclude that we may change r to any r' satisfying $r < r' \leq 64$.

10 Summary and Conclusions

In this paper we considered Blockchains as a well known mechanism for the creation of trustworthiness in a distributed network, as pioneered in the Bitcoin system [16]. Hereby, we introduced the distributed individual and layered wallet architecture as a means to store medical records in different levels of aggregation, in order to fine grain the interaction models according to the risks of personal data exposures and unique identifications. Simultaneously this enables a completely new scheme of patient to patient and patient to institution interaction with much high levels of trust and more data available to research studies.

As such, this medical wallet architecture allows for forum interactions based on anonymous hash identifiers but verified medical conditions to achieve a patient to patient supportive information trading. Further, this system offers the possibility for medical institutions, such as hospitals, to purpose much wider as well as more fine grained data analyzes by directly approached suitable patient groups also on an anonymous but verified basis. Hereby, such organizations profit from the fact that the approach integrates a functional access control structure, supported by Know Your Customer policies of integrated legal entities, which may even interact on a group basis. Moreover, patients have the opportunity to integrate further aggregation level, next to the level of raw data, for specific purposes, such as the communication with pharmaceutical corporations. Such companies may only require variations in the raw data, instead of the actual data, in order to pursue their drug studies, while they have the possibility

to directly reward the participating patients. This has the aim to significantly reduce their time of building study groups, while having more trust in the data. Patients thereby benefit from directly being able to trade their data, while having full control over granting and withdrawing as well as logging the individual access to their data. This can be also beneficial when changing medical professionals in cases of lost trust or, e.g., moving to a new city.

Furthermore, we demonstrated the formalized models of our architecture, including the medical record storage system, using IPFS, the key management and the actual Blockchain system. The latter is thereby a crucial part of the system, as we need to ensure the highest possible stability and system security when dealing with medical records as highly sensitive data. Therefore, we studied how Blockchains, and the choice and operation of cryptographic puzzles that drive the creation of new blocks, are controlled and owned by one or more organizations. Our proposal for such governed and more central control is that puzzle solvers are mere resources procured by those who control or own the Blockchain. This means that we not only control the accessibility of nodes but also of miners, in order to optimize the system stability according to specific environmental requirements as well as a financial budget, mostly influenced by energy and hardware cost.

We developed mathematical foundations for specifying and validating a crucial part of a governed blockchain system, the solving of cryptographic puzzles – where we focussed on Proof of Work. In our approach, owners of a blockchain system can specify allowed ranges for the size of the shared nonce space, the desired level of difficulty, and the number of miners used; and they can add mathematical constraints that specify requirements on availability, security, resiliency, and cost containment. This gives rise to MINLP optimization problems that we were able to express in analytical form, developing a *mining calculus*, by appeal to the ROM model of cryptographic hash functions used for cryptographic puzzles, where robust optimization models resiliency.

Based on this, we expanded the model towards the optimization of mining groups with different participating organizations as well as types of mining resources. This model allows us to suggest an optimal distribution of computational power between organizations of a consortium, e.g. health insurers.

These information as well as mathematical models inform now the development of machine learning-based engines, as the most crucial adaption of the permissioned Blockchain architecture. Hereby, we apply deep reinforcement learning to learn as well to decide about the optimality conditions at run time and to influence the immanent parameters of the Blockchain. This allows us to dynamically increase the stability and security of the system within financial borders as well as to apply further advanced, such as intrusion detection schemes.

We hope our work will provoke more thinking about the design, implementation, and validation of Blockchains that are controlled in a federated manner and that may fulfill domain-specific needs for the creation of trustworthiness. We believe that many domains have such needs that the approach advocated in this paper is able to meet the

inherent needs of advancements in medical data analytics as a patient-driven approach.

References

[1] Ali, R., Barrdear, J., Clews, R., and Southgate, J. Innovations in payment technologies and the emergence of digital currencies. *Quarterly Bulletin* (2014). Published by the Bank of England.

[2] Ben-Tal, A., Ghaoui, L. E., and Nemirovski, A. *Robust Optimization*. Princeton University Press, 2009.

[3] Bird, R. *Thinking Functionally with Haskell*. Cambridge University Press, 2015.

[4] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. SoK: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015* (2015), pp. 104–121.

[5] Castro, M., and Liskov, B. Practical byzantine fault tolerance. In Proc. *OSDI'99*, pp. 173–186, 1999.

[6] Danezis, G., and Meiklejohn, S. Centrally banked cryptocurrencies. *CoRR abs/1505.06895* (2015).

[7] del Castillo, M. The DOA Hacker is Getting Away. Online article on `coindesk.com`, 8 August 2016.

[8] Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., and Capkun, S. On the security and performance of proof of work blockchains. In Proc. *ACM CCS'16*, pp. 3–16, 2016.

[9] Horne, D. Hash chain. In *Encyclopedia of Cryptography and Security, 2nd Ed.* Springer, 2011, pp. 542–543.

[10] Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., Eds. *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. 2010.

[11] Kokoris-Kogias, E., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., and Ford, B. Enhancing bitcoin security and performance with strong consistency via collective signing. In Proc. *USENIX Security'16*, pp. 279–296, 2016.

[12] Lamport, L. Paxos made simple. *ACM SIGACT News 32*, 4 (2001), 18–25.

[13] Lundbaek, L., D'Iddio, A. C., and Huth, M. Optimizing governed blockchains for financial process authentications. *CoRR abs/1612.00407* (2016).

[14] Milne, R. Sweden's Riksbank eyes digital currency. Online article of the Financial Times, 15 November 2016.

[15] Misener, R., and Floudas, C. A. ANTIGONE: Algorithms for coNTinuous Integer Global Optimization of Nonlinear Equations. *J. Glob. Optim. 59*, 2-3 (2014), 503–526.

[16] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System, May 2008. Published under pseudonym.

[17] Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, 2016.

[18] Tawarmalani, M., and Sahinidis, N. V. A polyhedral branch-and-cut approach to global optimization. *Math. Program. 103* (2005), 225–249.

[19] Vigerske, S. MINLP Library 2. Online benchmark repository at `http://www.gamsworld.org/minlp/minlplib2/html/`.

[20] Vigerske, S. *Decomposition in Multistage Stochastic Programming and a Constraint Integer Programming Approach to Mixed-Integer Nonlinear Programming.* PhD in Mathematics, Humboldt-University Berlin, 2012.

[21] Vukolic, M. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In Proc. *iNetSec 2015*, pp. 112–125, 2015.

[22] Wattenhofer, R. *The Science of the Blockchain.* Inverted Forest Publishing, 2016.

[23] Zipf, M. *SAP HANA im Kampf gegen Krebs.* http://news.sap.com/germany/sap-hana-kampf-gegen-krebs/, 2014, accessed 01 August 2017.

[24] Zyskind, G., Nathan, O., and Pentland, A. Decentralizing privacy: Using blockchain to protect personal data. In Proc. *SPW 2015*, pp. 180–184, 2015.

[25] Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. thesis, Cambridge University

[26] Watkins and Dayan, C.J.C.H. Q-Learning Machine Learning, 8, 279-292.

[27] Mnih, V. and Kavukcuoglu, K. Methods and apparatus for reinforcement learning Google Patents, 2017 `https://www.google.com/patents/US9679258`

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller Playing Atari with Deep Reinforcement Learning DeepMind Technologies, 2013 https://arxiv.org/pdf/1312.5602.pdf

[29] Tambet Matiisen Demystifying Deep Reinforcement Learning http://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/, 2015, accessed 09 August 2017.

[30] Stanford University Stanford CS234: Reinforcement Learning http://web.stanford.edu/class/cs234/images/header2.png, accessed 09 August 2017.

[31] Tesla, Inc. Blog: All Tesla Cars Being Produced Now Have Full Self-Driving Hardware https://www.tesla.com/sites/default/files/images/blogs/ap_2_header.jpg, accessed 09 August 2017.

[32] Wikipedia, the free encyclopedia Example of a simple MDP with three states (green circles) and two actions (orange circles), with two rewards (orange arrows). https://en.wikipedia.org/wiki/Markov_decision_process, accessed 09 August 2017.

[33] NuCypher NuCypher KMS: Decentralized key management system (2017) .

[34] NuCypher Modern data security for hadoop (2017) .

[35] Pairing-Based Cryptography Proxy Re-encryption Systems for Identity-Based Encryption (2007) .