WHITE PAPER

Dynemix cryptocurrency platform

and

Liberdyne messenger

Dynemix is a blockchain platform of the next generation designed to become the first worldwide-adopted cryptocurrency capable of competing with conventional consumer-level payment infrastructure on equal footing.

Liberdyne is a P2P secure messenger powered by the Dynemix platform. Liberdyne is designed to compete with popular centralized messaging solutions on equal footing while at the same time offering users a superior level of security and privacy.

Both components of the project were developed in conjunction and are interrelated and mutually beneficial, which allows us to introduce features previously unavailable on both the cryptocurrency and instant messaging markets.

The following white paper contains an informal description of the project. The provided specifications are not final and are subject to amendments, if necessary, up to the mainnet launch. Any updates to this white paper can be made without prior notice.

The following white paper does not contain any token purchase offerings or any other information on the initial token distribution.

www.liberdyne.com

Contents

I. Background

In 2009, the first modern decentralized cryptocurrency, Bitcoin, was created. It was based on a distributed ledger called a blockchain.

In 2014, a fundamentally new blockchain platform – Ethereum – was born. Unlike Bitcoin, it was not strictly a cryptocurrency and was not primarily meant to become just a way of payment or a storage of value, but rather to be a decentralized virtual machine (also often called a distributed Turing machine, since its state machine used the Turing-complete instruction set) for running various kinds of applications, including so-called "smart contracts" that allowed value transfers and the automatic enforcement of other various types of obligations under multiple conditions. Native ETH tokens of the platform were considered a "fuel" for these applications, and blockchain was used as a database for storing applications as well as all concomitant data (accounts, messages, transactions, states etc.).

As Ethereum offered many new ways of using blockchain technology, while at the same time retaining all the main capabilities of Bitcoin, it was called a "blockchain 2.0" platform. We believe such a classification to be incorrect, however.

The problem is that, in our opinion, Ethereum and other decentralized virtual machine platforms should not be considered the next versions of Bitcoin-like payment platforms. They are, rather, separate parallel branches of decentralized ledger technology (DLT) and are connected horizontally, but not vertically.

One may disagree with us and say that if Ethereum has all the functions that Bitcoin can offer, plus a lot more, it undoubtedly should be considered just the next version of Bitcoin.

Though at first sight this statement may seem correct, actually it is the same as claiming that a motorbike is an improved version of a bicycle. Indeed, it has an engine and can go much faster, but only until the fuel runs out. No one would try to state that motorbikes should completely replace bicycles, despite the fact that they are both two-wheeled vehicles and motorcycles are obviously much more advanced.

The same approach can be applied to systems based on blockchain technology – though decentralized virtual machine systems can perform many more functions than decentralized currency systems, they are not different versions of one concept, but rather two independent concepts, as they are optimized for completely different uses.

For this reason, we believe that we should classify current DLT systems in a different way – what is commonly called a "blockchain 1.0" is rather a "decentralized currency system 1.0" (i.e. Bitcoin, Lightcoin and others), and what is commonly called a "blockchain 2.0" is a "decentralized virtual machine 1.0" (i.e. Ethereum), and most attempts to create a so-called "blockchain 3.0" are "decentralized virtual machines 2.0."

If we adopt this classification, it raises a question – what do we know about a "decentralized currency system 2.0"? Does it even exist? Indeed, there were some successful attempts to improve on Bitcoin architecture, which mostly concentrated on providing more anonymity and led to the creation of Monero, Zcash and several other platforms, but none of them made a step forward significant enough to be called a next-generation system. This is mostly due to the fact that, after the Ethereum release and its erroneous consideration as blockchain 2.0, developers abandoned the notion of a currency system and started working only on Turing machine platforms. This is quite logical, because why would someone develop the previous version of something when the next version is already on the market?

The consequences of this incorrect classification can be clearly observed at present. We still have no decentralized platform that can truly accomplish the initial goal of cryptocurrency – actually being a convenient currency. We consider this circumstance to be one of the main reasons blockchain technology has not become widespread. In fact, the crypto community today is a rather closed system of professionals and enthusiasts, growing at a fairly moderate rate.

The absence of such a platform leads to a drop in interest in technology among the population. We are already witnessing prolonged market stagnation, and we strongly believe that the deployment of a next-generation cryptocurrency system, which can compete in practice with conventional means of payment, is the only way to make a breakthrough in the market and finally spread the technology on a global scale.

In the following white paper, we are going to prove our ground and describe a next generation decentralized currency system capable of reaching the stated goal and becoming the first widely adopted mass market cryptocurrency.

The following white paper is not meant to be used for academic purposes and does not adhere to academic style or methodology. Instead, we provide an informal description of the project, making it easier for a general audience to comprehend the design of the system.

II. Problem Statement

1. Creating a decentralized currency system 2.0

Since we intend to create a decentralized currency system 2.0, we should firstly define what exact properties such a system should obtain, and we will instantly see why it has nothing to do with the notions of so-called blockchain 2.0 or blockchain 3.0 projects, as well as why Bitcoin and other so-called blockchain 1.0 projects are not capable of obtaining these properties.

Essentially, we can outline two main features that we want to see in a perfect decentralized payment system: it should be actually decentralized, and it should provide a user experience similar to existing centralized payment systems (or even better, if possible) to be able to offer serious competition.

In our opinion, the properties required to accomplish these stated tasks are as follows:

1) Guaranteed instant transaction processing

In conventional centralized payment systems, any transaction sent to the network will be instantly processed by the operator, except when it does not fit the requirements (e.g. if the transaction amount exceeds the user balance) or a fault occurs. We should achieve the same properties in our blockchain system and design it in such a way that it can assure that all currently pending transactions are added to the next block.

Blockchain 1.0

In Bitcoin, transactions to be included in a block are selected by miners of their own will. There is no guarantee that any particular transaction will be ever included in the blockchain. As conceived by the creator, miners are incentivized to add transactions to a block by obtaining the commissions included in each transaction. The protocol, however, contains no strict rules about transaction selection; hence, miners are free to produce even [empty blocks] if they wish.

Blockchain 2.0

Ethereum has the same rules of block creation as Bitcoin. This turned out to be a serious problem when in Q4 2018 we witnessed certain mining pools creating empty blocks (so-called spy mining). There is currently no solution for a possible censorship issue either.

Blockchain 3.0

Mostly the same situation can be seen in all widely known blockchain projects to date. Only the developers of DAG-based systems claim to provide processing of every transaction, but their design relies on altruistic behavior by participants, so it cannot be considered a proper solution to the problem in a Byzantine environment. Some blockchain developers also addressed the problem (e.g. Honey Badger), but no one fully provided the stated properties.

2) Quick finality

There should be no scenarios where users have to wait more than 10–20 seconds for a transaction to be completed. In centralized systems like Visa and MasterCard, it usually takes just a few seconds (although we should note that, even being confirmed, the transaction takes a lot more time to be finalized, but this does not significantly affect the consumer experience), so not many users will agree to wait much longer.

Due to the distributed system's specificity, we cannot match the speed of centralized systems (because we need to use a consensus protocol, which takes time to process and synchronize the data between peer nodes). We must, however, get as close as possible.

Blockchain 1.0

In Bitcoin, a transaction that has been put into the blockchain is considered to be reliably confirmed after six blocks have been mined on top of it. This takes more than an hour, which feels like an eternity in comparison to credit cards.

Blockchain 2.0

Ethereum has a much shorter block time (about 15 seconds), which is why its confirmations occur faster. However, Ethereum requires more confirmations to consider a transaction finalized with the same level of reliability as Bitcoin's six confirmations. Generally, reliable finalization is reached within about 1–3 minutes, which is still unacceptable in terms of competition with centralized systems.

Blockchain 3.0

Processing speed is one of the parameters on which most developers concentrate, so the majority of current-generation projects claim to reach finality latency close to that of centralized systems or even surpassing them.

Although the Byzantine fault tolerant (BFT) consensus model, which is used in most recent protocols, allows reliable finality to be reached quicker, most of these claims are based on heavily centralized system architecture. Nevertheless, matching our targeted latency boundary is feasible for many recent BFT-style designs.

3) High transaction processing rate

According to Visa, VisaNet is now processing an average 1,700 transactions per second (TPS). If we want to create a system that can be as widely used as conventional means of payment all around

the world, it should be capable of handling about 10,000 TPS. Moreover, this should be achieved within the main transaction database without second-layer technologies like payment channels.

Blockchain 1.0

Bitcoin can process about 7 TPS the with the current block size, which is extremely insufficient. Though this is nothing but an arbitrary choice, in any case, Bitcoin's architecture cannot provide thousands of TPS on-chain with an acceptable decentralization level. Developers try to implement second-layer solutions (such as Lightning Network), but in our opinion it feels more like concealing the problem rather than solving it.

Blockchain 2.0

Ethereum can provide twice as much TPS as Bitcoin does, but this is not a significant improvement. There are also second-layer technologies (e.g. Raiden, Plasma), but generally, from the perspective of scalability, Ethereum has not improved much on Bitcoin. The situation will significantly change only when Ethereum 2.0 is finally presented.

Blockchain 3.0

Since scalability has recently become the unofficial primary benchmark of a blockchain system's advancement level, developers started a contest for the highest TPS claim. Most of these claims have nothing to do with real-life scenarios, as they are based either on the capabilities of second-layer solutions (which are theoretically limited only by hardware and the bandwidth of all nodes combined), or they do not take into account the actual possible hardware and bandwidth capabilities of peer nodes in a practical environment.

A realistic throughput boundary that can be practically achieved by non-sharded designs with the current state of hardware and communication infrastructure lies roughly between 1,000–5,000 TPS and can be pushed further only with the help of sharding.

4) Free transactions

In centralized systems like Visa and MasterCard, commissions are charged to payment-system participants, but ordinary consumers can perform most operations free (consumers usually pay banks only for cards issued and account maintenance). Therefore, for most users, transactions seem to be free of charge. Only those few who actually know how the system works understand that the vendor includes a transaction fee in the price of a service or good.

A free-of-charge payment system can become very attractive both to users and to businesses, especially for microtransactions. The absence of commissions can give it a very strong advantage over other payment systems of either centralized or decentralized design.

Blockchain 1.0

Bitcoin has rather high transaction fees, which grow even higher during periods of increased system load (for example, in December 2017). This may be partly caused by the Bitcoin Core developers' refusal to increase block size. This situation has long been discussed by the community, but we still see no changes. Other altcoins of similar design will most likely face the same situation if they become similarly popular. The only popular platform that offers free transactions is IOTA, which is a DAG-based system.

Blockchain 2.0

Ethereum has lower fees overall than Bitcoin. This can be explained by its much lower market cap and popularity, combined with its different approach to block sizing. Since it is a Turing-complete distributed state machine, transaction fees depend highly on the complexity of the app that issues the transaction. Moreover, for this reason, a transaction fee consists of two parameters: gas price and gas limit, which should be defined by the transaction issuer. This makes the system a little less clear for common users and negatively affects their user experience.

Blockchain 3.0

Many projects claim to be able to maintain extremely low transaction fees, but these statements cannot be verified, since none of these projects have reached a system load close to that of Ethereum. What can be said for sure is that, being virtual machines (although not always decentralized), they all have the floating commission issue that is intrinsic for platforms of this type.

5) Financial privacy

Since blockchain is a public ledger, any person can view all of the transactions in the network. Hence, if the user discloses his or her ID to a third party, this party will instantly know the user's current balance and entire financial history. This is a serious flaw from the point of view of an average consumer who intends to use such a system for everyday payments, which means that the system should be capable of hiding transaction attributes and balances from the public.

Blockchain 1.0

Surprisingly, first-generation projects are the only ones among popular platforms that offer banking secrecy compatibility, but this fact has a reasonable logic beyond it – the technical solutions offered are applicable only to UTXO based platforms and put an intense load on the system, which is why they are less compatible with the decentralized virtual machine concept. The most popular platforms that offer a means of hiding transaction attributes are Monero, Dash and Zcash.

Blockchain 2.0

Ethereum does not offer any transaction attribute-hiding solutions on the protocol level. Although there are projects devoted to bringing financial privacy to Ethereum via smart contracts, any significant steps toward this goal on the platform scale will not be introduced any sooner than Ethereum 2.0 is presented.

In any case, since Ethereum uses an account model and focuses on smart contracts, it cannot use any currently available privacy solution.

Blockchain 3.0

Not many developers of recent projects care about this factor, and, surprisingly, some even state that anonymity should be excluded from cryptocurrency operations. We can state that no better solutions than the ones used in the first-generation blockchains mentioned have been introduced to date.

6) True decentralization

We already have a working centralized payment infrastructure, and there is no need for another similar system that differs only in the presence of the word "blockchain" in the title. Since making

the system more centralized is apparently a way of resolving the scalability issue, many developers have chosen this path.

We do not support this approach and believe that decentralization is a crucial feature, which is why the system should be designed in a way that prevents any possibility of oligopolistic control and censorship attempts or any other possible kinds of coordinated selfish behavior.

Blockchain 1.0

Bitcoin was the first working concept of a decentralized payment system, and its creator paid a lot of attention to the notion of decentralization. The very reason of Nakamoto consensus invention was the need to find a fault-tolerant and Sybil-proof solution for setting transaction timestamps without involving a trusted intermediary.

Although the proposed solution seemed appropriate in the early stages, the appearance of mining pools dramatically changed the situation, and currently the major fraction of Bitcoin's hashrate is controlled by just a few actors. The oligopolistic trend shows that the PoW concept is far from perfect in terms of providing decentralization, not even mentioning other known issues, like AsicBoost.

Blockchain 2.0

Ethereum has the same issues overall as Bitcoin, since they use very similar consensus protocols. Transition to the PoS protocol is scheduled in Ethereum 2.0, but we cannot reasonably evaluate it until we see its final version implemented in the mainnet.

Blockchain 3.0

Most developers have concentrated on scalability and speed issues, trying to provide as many TPS as possible. The easiest way to speed up a decentralized system is to centralize it (thereby reducing its security as well), which is the path most projects followed. We can see a clear trend of moving back to something that looks more like client-server architecture, while pretending to be a peer system.

That has led to the creation of pseudo-decentralized systems, where instead of one trusted processing party, blocks are formed by a limited group of pre-appointed validators, thus making these systems not much less centralized than Visa or MasterCard.

To date, we can state that the overall situation with decentralization among the majority of recent platforms is very sad, despite the fact that the decentralization concept is actually the very reason for cryptocurrency technology's existence.

7) Widely distributed system support

One of the key factors influencing the level of decentralization is the number of nodes that participate in a state transition. The system should be designed to let the maximum number of users be involved in the system's support processes: in other words, we should distribute trust as much as possible.

Distributing rewards among a higher number of recipients also benefits the economy of a cryptocurrency, providing more independence (since it prevents the concentration of large numbers of issued tokens within a limited group of users) as well as stimulating the growth of a real economy.

Blockchain 1.0

The essential properties of the protocol seemed to fit the concept described in its early stage, but the hashing algorithms used in Bitcoin and its successors allowed the creation of ASICs, which, together with a constantly growing level of difficulty, made mining a purely professional activity.

Although any person could theoretically participate in mining by joining a major mining pool, cost optimization requirements have become a serious obstacle for amateurs.

Blockchain 2.0

Ethereum uses hashing algorithms that work optimally on GPUs (although the ASIC problem is still not completely resolved). This makes it easier to participate in mining with quite standard hardware. The problem of cost efficiency is still present, however, as it becomes an essential companion of any PoW system once it reaches maturity. The current situation shows that we need a different state transition algorithm if we truly want to preserve widely distributed system support and resist mining professionalization.

Blockchain 3.0

As we mentioned, most developers have concentrated on the speed and scalability of their platforms. The easiest way to enhance these parameters is to delegate system support to a highly limited number of validators with professional hardware, and this is what many recent projects tend to be like.

8) Economic potential

Existing cryptocurrencies, in their current state, remain mostly a speculative asset and do not transform into a medium of exchange for goods and services. Speculative manipulations greatly increase their volatility, thus further hindering the development of a real economy.

Besides implementing all the features mentioned above, which should make the system competitive with fiat currencies in terms of their user experience, it is also essential to provide a well-designed economic model that can help overcome the speculative phase.

Blockchain 1.0

Even though it was introduced by its creator as a decentralized payment system (literally “a peer-to-peer electronic cash system”), Bitcoin did not quite turn out to meet this definition, and became instead a speculative asset or an inflation hedging tool. We can conclude that from now on it will most likely remain digital gold, merely providing the same functions that precious metals do currently in the economy. We see no way for Bitcoin to transform from its current state into an actual cash system with the function of a universal medium of exchange.

Blockchain 2.0

The situation with Ethereum (ETH) is slightly different. Since Ethereum is used mostly as a platform for conducting ICOs, ETH tokens power the market of various tokens issued by third parties. Though most of these tokens have the properties of securities, and hence it is still mostly a financial market instead of a real economy, it actually serves the main purpose of the platform and provides token use apart from just speculative trade. In any case, as Ethereum was never particularly meant to be a currency, the current situation is justified by ETH’s initial purpose to be a fuel for Dapps.

All the same problems can be found in the newer platforms, since nobody has put enough effort into improving their economic aspects, concentrating instead mostly on their technical features (such as speed and scalability, which does not improve the economy much). This leads to the stagnation of most platforms right after their release, but developers usually do not seem to care much.

2. Current state of the industry

1) Brief summary on features we intend to achieve

Here is a brief summary of the properties that we find essential for an ultimate decentralized currency system:

a) **Decentralization.** The system should be resistant to oligopoly control, censorship attempts and cartel formation. As well, the functions of system support and rewards for performing these functions should be distributed among as much participants as possible (which means that the nodes should be capable of being run on common desktop-class hardware and have relatively low bandwidth requirements).

b) **Closeness to centralized systems' user experience.** The platform should process all transactions (despite any preferences of the block producers). Each transaction should be finalized within 10–20 seconds. The platform should handle 10,000 TPS with tolerable overhead, process transactions free of charge and provide financial privacy.

c) **Economic potential.** The platform should be capable of transitioning from the initial investment stage to the development of a real economy.

Having stated all of the basic requirements for a cryptocurrency that actually intends to become a fully-fledged currency in itself (something we may call a "decentralized currency system 2.0"), we can finally give a reasonable answer to the question: Are there any existing projects that match at least most of the requirements? The answer is no.

2) Bitcoin as currency

As we can see, Bitcoin, for example, fully matches none of the stated criteria:

- No impartial transaction processing is available, and miners are free to reject any transactions at their own discretion.
- It processes transactions very slowly: 60 minutes to a get reliable confirmation.
- It provides only 7 TPS.
- It features high commissions, especially during periods of increased load.
- It does not offer financial privacy. Every user can see all other users' histories of transactions.
- It is not decentralized enough, since a limited number of mining pools have obtained a very strong influence on the system.
- System support is concentrated mostly in the hands of professional miners who possess the most powerful farms.
- Its economy got stuck in the speculative stage.

It seems absolutely clear that, being this far from providing a user experience similar to common centralized payment systems, Bitcoin never had any chance in substituting for a conventional fiat-based payment infrastructure.

Although there are plenty of Bitcoin maximalists who are ready to tolerate any flaws of the platform and truly believe that it can replace the entire fiat finance framework, their numbers, unfortunately, are far from sufficient, and there are not many such people left uninvolved to date.

Those who prefer a more pragmatic approach may agree that decentralization is a very promising paradigm, but, at the same time, they will consider switching to it only after someone offers a platform that can provide the level of convenience comparable to conventional centralized systems.

And finally, the vast majority of people around the world just do not understand or care about the notion of decentralization. Their interest can be triggered only by simpler, down-to-earth advantages, such as fast and free transactions, fully anonymous payments, the possibility of mining some coins without large investments etc.

That is why Bitcoin's userbase will not scale up to hundreds of millions of active users. The platform's design just does not have enough to offer a general audience.

Does it mean that Bitcoin will inevitably fall? We do not think so. As we said, Bitcoin is capable of performing the same functions in the economy as gold currently plays.

Not long ago, gold used to be a medium of exchange and fully performed the functions of money. Eventually, more convenient means of payment were invented (banknotes and finally digital money, which were untied from gold at a certain point), and gold mostly lost its initial function. As we can see, it did not completely remove gold from the economy, but rather transformed its functions into that of an inflation-hedging tool.

Since Bitcoin has certain properties in common with gold and other precious metals (i.e. a limited supply with slow, controllable issuance) and its technical drawbacks do not influence its ability to perform the stated hedging function, it may find its place next to gold or even eventually replace it, as the world is constantly trending to digitalization.

One may notice that this conclusion does not fit our initial classification. In the beginning of this white paper, we stated that Bitcoin should be classified as a decentralized currency system 1.0. Indeed, having been called a peer-to-peer electronic cash system by its creator, it was conceived to be used as such. We believe, however, that at the current stage we should reconsider the classification of blockchain platforms. Our refined vision will be described in the chapter devoted to competition.

3) Other platforms as currencies

What about Ethereum and other decentralized virtual machines (so-called blockchain 2.0 and 3.0)?

Although implementing smart contracts on blockchain is a really great idea, it actually does not help a cryptocurrency to be a better currency. In reality it does exactly the opposite – it degrades the properties of a given blockchain system as a payment processor, since it applies complications that do not serve payment functions but rather make the platform a more universal tool.

As a completely different concept, it should not have been called blockchain 2.0, because it does not replace currency platforms. Both concepts can easily co-exist, and each can follow its own path of development.

After the appearance of Ethereum, the concept of Bitcoin-like payment systems was abandoned for good, and since that time we actually have not witnessed any serious attempts to bring about a next-generation blockchain system concentrated solely on payment functions.

We see two main reasons for this situation:

a) Developers do not try to create such a system because the "blockchain 1.0, 2.0 and 3.0" classification has been fully entrenched through the media. Developers must follow the rules and try to make some kind of a blockchain 3.0 system if they really want to attract sufficient funding. Therefore, most projects arrive under the slogan "make Ethereum, but faster." One

must be bold enough to go against this trend and convince the community of the falsity of this approach.

b) It is actually very hard to design a system that can embody all the stated parameters, since they conflict with each other. For example, it is commonly known that increasing system performance will inevitably reduce decentralization or security. Finding an optimal balance of all parameters and developing an architecture that will be a perfect compromise is a very hard task that nobody is willing to try to accomplish.

That is exactly why we are here: to fill this gap and introduce a decentralized payment system of the next generation capable of becoming the first mass-market cryptocurrency.

3. What is wrong with smart contracts?

Since we call the current classification incorrect, we should explain our point of view on decentralized application technology and show why this feature is not very beneficial for a blockchain payment platform and why we should avoid merging the two notions and finally recognize them as different concepts.

Actually, we like the concept of a decentralized virtual machine, and Ethereum as the first embodiment of this concept. The main issue is that, currently, its potential is highly overestimated. Let us explain why.

1) Smart contracts and their use cases

If we consider the classic notion of a smart contract, which existed long before the invention of blockchain (the first known concept was written about by Nick Szabo in 1995), we can roughly define a smart contract as an algorithmically self-enforcing contract.

When someone concludes a common free-form contract, for example to purchase a good, both parties must take actions of their own will to fulfil their obligations. A smart contract, on the other hand, can initiate enforcement after the occurrence of certain conditions predefined by the parties and automatically fulfill the contract without any human involvement.

The simplest example of smart contract implementation is a vending machine. When a buyer chose an item by pressing the appropriate button, a machine's firmware initiates a smart contract. It waits until the buyer puts enough cash into the bill acceptor and then dispenses the selected item, thus automatically fulfilling the seller's obligations without human involvement.

It can be easily concluded that smart contracts do not have a very wide scope of application. We see two use cases in which smart contracts can be engaged:

a) **They can be applied to goods and services that can be delivered via software algorithms.**

This means any goods and services that can be dispatched automatically (for example, trading from fully automated warehouses, automatic car washes, or any type of digital content) or that can be controlled via software-driven devices (for example, a car-sharing service with a mobile app).

b) **They can be applied to situations when a transfer of the title of ownership is enough to consider obligations fulfilled.**

The most common example is non-certificated securities – since they do not exist in material form, their transfer is conducted simply by making a record in the database.

2) Decentralized smart contracts and their limitations

Though we see that there are a limited number of smart contracts use cases, this technology looks promising, especially considering the current trend of digitizing business processes.

When we put smart contracts into a decentralized environment, however, the situation changes dramatically. The problem is that the rules of decentralized systems further limit the already limited smart contracts' capabilities.

a) **Decentralized smart contracts are completely independent.**

While centralized smart contracts can play an ancillary role in traditional legal agreements and execute certain predefined parts of parties' obligations, a smart contract put into a decentralized environment cannot refer to any off-chain agreements, and all rules for interactions between the parties should be coded within a smart contract (or a set of smart contracts within one blockchain).

While centralized smart contracts can interact with different sources, decentralized smart contracts can interact only with other smart contracts within a given decentralized environment and cannot process data from any off-chain sources, since contract instances are replicated on multiple nodes and, in the case of operating with non-deterministic functions upon receiving non-identical data, the consensus may be broken. This issue may be solved by trusted oracles, but introducing a trusted intermediary into a system that was designed particularly to avoid trusted intermediaries contravenes the initial idea behind decentralization.

b) **Decentralized smart contracts are outside the legal field.**

In most countries, digital agreements are recognized as legitimate (although an expert opinion is often required to assess the content of the contract and the circumstances of the dispute), so when enforcement algorithms of a smart contract fail or a dispute between the parties occurs, standard legal procedures can be initiated (for example, filing a lawsuit).

The legal status of decentralized smart contracts is highly questionable. Although, technically, we can apply the same regulations to them as to centralized contracts, it is still unclear how any external source would be capable of influencing the execution of a decentralized smart contract, since one of the main ideas behind decentralized systems is to resist any attempts at censorship.

This means that standard legal procedures are not applicable in this case. Considering that one of the distinguishing features of law is that it is backed by the coercive power of the state, smart contracts are placed out of the jurisdiction of law enforcement. That is why decentralized systems are considered to exist within the unique "code is law" paradigm, and that is why we cannot actually apply the terms "law" and "legal" to any rules contained within a decentralized environment.

One may disagree and state that we already have a few platforms with governance, but the problem is that governance models may violate basic cryptocommunity principles and make these systems centralized. For example, we cannot call EOS a cryptocurrency, for reasons including its governance model and other features of the platform's architecture that rely heavily on centralized maintenance.

c) **Decentralized smart contracts are immutable and irreversible.**

One of the distinguishing features of blockchains is impossibility to make changes to the database history. Hence, once a smart contract is added to a blockchain, it cannot be stopped, altered or terminated (unless particular conditions are specified in the code), and any actions performed by it are irreversible (again, we cannot refer to platforms like EOS, which actually allow the reversal of an already-confirmed transaction, because such a feature just shows that the system is not decentralized and that we cannot call it a proper blockchain system).

So, if a developer has not foreseen certain problematic situations while writing a smart contract, mistakes cannot be corrected afterward, and if there is a bug in the code, the consequences can be even more terrible. As we all remember, The DAO contract bug aftermath could only be resolved by a hard fork.

Taking into account all these issues, we can conclude that, though we can propose many potential use cases for decentralized smart contracts, when it comes to actual implementation we will likely face so many obstacles that such smart contracts may turn out to be absolutely unusable in many real-life scenarios.

Let us consider one example to which people often refer when intending to demonstrate the potential of decentralized smart-contract technology – a land registry. It is stated that we can move a land registry (or any other title-registry system) to a blockchain via a smart contract and perform all kinds of deals with registered property directly on-chain. It sounds very promising, but the truth is that we cannot actually do this.

One reason is that land is an object of both private and public interests. Such circumstances as target usage, intensity of maintenance and land productivity are monitored by authorities, and administrative measures can be applied in case of the detection of violations. These measures can be applied, including in the form of deprivation of the right of ownership, which necessitates making a record in the land registry. In the case of registry keeping within a blockchain, it leads to one of two unacceptable situations: either we provide a third party (e.g. specially appointed arbiters, judges, government agencies etc.) with the opportunity to make records in the registry against the will of title holders, hence violating the decentralization principles, or we lose the opportunity to protect public interests.

What if we take other types of property that do not involve public interest? This will not solve the problem even for a bit. There are myriad situations concerning property deals that we cannot script in a smart contract. All regulations concerning the invalidity of contracts, inheritance, bankruptcy procedure etc. become inapplicable. This means that, by moving any kind of title registry to a decentralized blockchain, we are not only changing the underlying technology but also disrupting the entire legal framework of property relations.

Another problem is privacy – blockchain is a public ledger, which means that any person can access all the data stored on the blockchain, including the personal data of the property owners. We can store data in an encrypted form, but then it raises the questions of key storage and data access.

Eventually, we will conclude that we are going to need a private blockchain administrated by a single authority to satisfy all the required conditions, and this means that we simply do not need blockchain for this, as none of the key features of blockchain can be properly used.

Frankly speaking, these problems are also relevant to common blockchain payment systems like Bitcoin – an inability to impose a penalty on the funds seriously affects legal relations as well, and privacy is also a relevant problem, but we may state that in this case the cons are balanced by the pros, while the case of smart contracts brings too many cons.

3) Requirements for unlocking smart contracts' potential

If we want to build a balanced, decentralized smart-contract platform that could be applicable to various real-life scenarios and thus capable of taking full advantage of smart contracts' potential, we should meet the following conditions:

a) **We need to codify the law for a decentralized environment.**

   And by "codifying," we mean literally to codify – to transition common legal norms into software code. Since no conventional legal system is fully applicable to a decentralized environment, we will have to alter many laws and even basic principles of law, thus creating an absolutely new and unique legal framework.

   This is a tremendous task, considering that traditional legal systems have been developing for millennia. Even after we manage it, the new system will suffer from an infinite number of flaws and gaps that will keep it unusable for decades until most of them are revealed and eliminated.

b) **We need to appoint an arbiter to resolve disputes.**

Of course, having developed a codified legal system, we can resolve certain generic situations without an arbiter with the help of the scripted rules, but there are too many variables to predict every possible situation. Considering that any gaps will be heavily abused, we cannot rely solely on scripts.

Due to the nature of decentralized systems, we cannot appoint a human arbiter, so the optimal solution is to involve an AI. Therefore, we need to develop an AI with technical features that correspond with the principles of cryptocommunity and make it capable of understanding our coded laws and the facts of the dispute, then solving the dispute correctly. Hence, we have another incredibly difficult task.

c) **We need to adopt an off-chain system capable of establishing legal facts in a form understandable by AI.**

Any legal dispute settlement consists of resolving two components: a Question of Fact and a Question of Law. Even if the latter can be resolved by a combination of a scripted blockchain legal system and an AI, a Question of Fact will not always be resolvable with on-chain data alone. Many situations will require human involvement. Therefore, we need to adopt a specific legal framework and a technical protocol to allow human–AI interaction on the matter.

This brings us back to the problem of trusted oracles, and this issue may likely become the bottleneck of our system's decentralization.

To date, there are several concepts of decentralized oracles based on games with focal points, in which participants reach consensus by voting for every value fed. Although generally efficient, such solutions may fail at a certain point, when the disputed value grows high enough to make corrupting attack vectors eligible. In addition, they are only efficient when the value is publicly known, but they cannot provide a robust solution to situations in which the value can be generated arbitrarily by a single source or a limited number of sources.

Overall, it seems clear that it will take not years but decades before we will be able to take full advantage of the potential of decentralized smart contracts. So far it looks like a promising technology, but not one that can be fully used right now.

If we look at the current situation, this statement is mostly confirmed by experience. Ethereum has been out for enough time to evaluate it, including how its smart contract potential was used by the developers.

We have seen three major methods of using the Ethereum platform:

- ICOs
- DeFi
- CryptoKitties.

Despite the widely discussed potential of smart contracts, we have still not seen many real-life cases in which the smart contract technology provided any significant breakthroughs (except maybe gambling). We can roughly describe Ethereum in its current state as a platform powering securities exchange and crowdfunding.

4) Cost of smart contract implementation

Now that we actually understand what the implementation of smart contract technology can bring to a blockchain platform, let us evaluate what price we would have to pay for it.

In the beginning of this chapter, we identified the set of properties we want to achieve in creating a next-generation cryptocurrency platform. The problem is that implementing smart contracts will impede some of the system's stated properties:

a) **Free transactions.**

Since user-generated transactions in a blockchain payment system are simple token transfers from one address to another, it is easy to evaluate how much traffic and computing power will be consumed by each transaction. Therefore, we can apply uniform rules to all transactions.

If we add smart contracts, the situation changes dramatically. Every smart contract is a separate application, which should be processed by nodes in order to complete a transaction based on the smart contract. Since smart contracts have different levels of complexity, the loads put on the system while the transaction is processed will also differ. Some smart contracts (e.g. those containing endless cycles) will be able to overload the system easily, so it is necessary to charge fees every time a smart contract is processed.

Therefore, smart contracts eliminate the concept of free transactions, one of the parameters we identified as intending to achieve. It is possible to make separate rules for common transactions and those issued by smart contracts, but this may create turmoil for most users who will not be able to distinguish among the different types of transactions.

b) **Distributed block production.**

One of our key goals is to decentralize the platform to the maximum degree possible. To achieve this, we must design the system in such a way that, even under a very intense load, full nodes (capable of block production) could be run on a common laptop or desktop PC available to the average user. At the same time, the traffic overhead should not exceed the bandwidth available to the vast number of users.

We may be able to meet these restrictions, but only when we know how much load each transaction will generate. In the case of smart contracts, the system load becomes rather unpredictable, and we risk exceeding the required parameters, which will lead to nodes' complete migration to data centers as the system scales up. This is a highly undesirable scenario and another serious reason to reject the idea of smart-contract implementation.

c) **Reliability of the system.**

Implementing virtual machine technology dramatically increases the number of possible vulnerabilities and bugs. Every existing smart contract platform has had to come (or has yet to come) a long and painful way to reach relative stability and security. Even Ethereum, which was introduced five years ago, still encounters various security issues.

If we presume that we are going to develop a potentially popular and widespread currency with a high circulating volume, then it is crucial to make the system as reliable as possible. Implementation of a technology as complicated as smart contracts would definitely not help us reach this goal, especially taking into consideration our intention to comply with financial privacy standards, which imply balance encryption and hence obstruct the detection of violations.

d) **Real-economy orientation.**

Native tokens of smart-contract platforms perform specific functions, serving in the first place as fees for the computations performed by block producers. Considering their embedded value, they can also serve as a medium of exchange or value storage, but performing these functions is not their primary purpose.

Meanwhile, creating a platform with tokens that are primarily meant to be used as a universal medium of exchange requires a specific approach to economy, which will not be optimal for a smart-contract platform.

If we admit that government bonds, for example, can be used instead of a currency for settlements between certain major players on the market, this does not mean that they can

replace cash on the consumer level. In the same way, tokens that serve as a fuel for Dapps cannot replace currency tokens, as they cannot bear the optimal economic properties.

Considering all the provided analysis and keeping in mind our primary goal (to create a payment system with native tokens that can be used as a universal medium of exchange), we can conclude that building a decentralized virtual machine does not coincide with our vision, which is why we believe that we should leave this feature to Ethereum and concentrate on features that really matter: free transactions, true decentralization, reliability and an optimized economic model.

This is also why we should stop calling smart-contract platforms blockchain 2.0 and blockchain 3.0. We clearly see that they are a totally different concept of blockchain technology use, but not an upgraded version of decentralized payment systems.

We should note that smart contracts we are talking about do not completely fit the definition of Ethereum's smart contracts. Since the Ethereum platform uses Turing-complete languages, any kind of app can be coded on the Ethereum blockchain, including, but not limited to, smart contracts. For example, nothing can stop someone from creating a Super Mario game on the Ethereum platform. It will be a super-slow, super-expensive Super Mario, but it is still possible. From the point of view of Ethereum terminology, such an app would be still powered by a smart contract, even though we see that it has nothing to do with the notion of smart contracts we described earlier. Ethereum developers did not use the term very aptly, thus creating a bit of confusion.

Regardless, considering that Ehereum Virtual Machine's computations are extremely slow and expensive, we may conclude that it is suitable mostly for scripts that involve value transfers, hence smart contracts or at least parts of them. Even games built on the platform usually operate with tokens that have some material value. This means that our definition of smart contracts, though not strictly matching Ethereum's terminology, may be practically applied to the subject at hand.

4. Overcoming the crypto community's entry threshold

After we create a system that is perfectly capable of being a consumer-level payment platform, we will face another problem: how to spread such a system among a wider audience.

As already mentioned, the crypto community today is a rather closed system of professionals and enthusiasts that grows at a moderate rate. One of the main obstacles to the rapid growth of the number of people involved in the crypto community, in our opinion, is a high entry threshold caused by the following factors:

a) **The complexity of use**

Very specific knowledge is required to use the existing tools. Not all IT professionals, to say nothing of inexperienced users, understand the mechanisms of cryptocurrencies (cryptowallets, public and private keys, light and full clients, smart contracts etc). Ordinary users lack access to a simple and understandable product that would allow for the use of cryptocurrency in two clicks.

b) **The need to invest fiat funds**

To obtain crypto coins, an ordinary person first needs to understand how the exchange service works and second needs to spend a certain amount of fiat money. One may try to obtain cryptocurrency by mining, but nowadays mining has turned into a professional activity that is absolutely impractical without special knowledge, experience and significant investment.

Considering that cryptocurrencies face strong prejudice created by media among the general audience, the risk of losing money mostly prevails over the audience's curiosity.

c) **The lack of real use scenarios**

Cryptocurrencies are mostly used as an investment tool, and the real economy within the cryptocurrency environment is just beginning to develop. The volatility of the exchange rate does not help the development of the real economy either, but this does not seem to bother many developers, who are well enriched from bubbles in the stock market. As for the users who are not interested in investing their savings, most cryptocurrencies cannot offer them any relevant methods of use.

As a result of these obstacles, many people unrelated to the IT industry refrain from using cryptocurrencies, despite the fact that they may have an interest in them. In fact, the money decentralization concept introduced by Bitcoin is beneficial for the general audience and not only for geeks, but there is too much confusion caused by presenting Bitcoin as a bubble and a financial scam by media. Combined with the other mentioned threshold factors, this heavily reduces the potential for cryptocurrencies to proliferate.

Having assessed all of these factors, we came to the conclusion that if we really intend to introduce a cryptocurrency as a payment instrument for the consumer market, we must present not a mere blockchain protocol but a complete product on top of the protocol with the possibility of integration into a convenient ecosystem. If we simply create a technical platform, we will face the same entry threshold that the users of other cryptocurrencies are forced to overcome. In this regard, for the cryptocurrency to accomplish this task, a basic platform app should be created that will be capable of becoming a popular product itself, gaining interest among users regardless of the presence of an integrated cryptocurrency.

A social network could be the ideal environment for the spread of a cryptocurrency. Social networks, however, have one property that is incompatible with the basic principles of cryptocurrencies: the need to share the user's identity directly or upload data that allow for the indirect identification. An anonymous social network is simply an oxymoron; however, an anonymous messenger is a pretty viable concept.

5. P2P messenger as a basic application for the distribution of cryptocurrency

Today, the most popular applications of this type are centralized messengers, which use servers to process most of the user's data (WhatsApp, WeChat, Viber etc.).

After the revelations of Edward Snowden, the problem of insufficient user privacy has begun to be actively discussed. To provide a solution to this problem, the developers of messengers have implemented end2end encryption.

This solves the issue of data privacy only partially, however. The main problem of popular messengers lies in their centralized architecture. No matter what encryption they use, there is always a person managing the servers. By applying pressure on this person, one could obtain control of the servers and therefore the user data stored there.

One may ask, if developers apply end2end encryption, what kind of data we are talking about, exactly? We see several problems concerning privacy in modern popular centralized messengers:

a) **End2end encryption does not necessarily prevent third parties from reading encrypted conversations.**

Firstly, in most cases we cannot be sure that developers do not have an opportunity to obtain private keys. Most messengers' source code is not publicly available, and those who claim to be fully open source and free might turn out not to be so open and free (for example, the available Telegram client source code is usually outdated, and no server-side code was ever published). Considering that client apps transfer a lot of metadata to servers, nothing stops developers from secretly obtaining private keys.

Secondly, even if all the transferred data is securely encrypted and no keys are sent to the developers, it is still possible to remotely read the whole chat history stored on the device. There are multiple reports from users who claim to be getting targeted ads on Facebook regarding keywords that they typed only in WhatsApp chats. Users are wondering how Facebook can know about the contents of their WhatsApp correspondence if it is totally encrypted.

It is actually technically possible via data sharing between apps. For example, in earlier iOS versions, each app had its own totally separated sandbox and no data sharing between apps was allowed. iOS 8 introduced a "shared container," which allowed particular data to be shared between apps in the same app group. The iMazing app developers recently discovered that the WhatsApp, Facebook Messenger and Facebook apps actually share the same container, called group.com.facebook.family. This, together with the mentioned user reports, may lead to certain conclusions. Since WhatsApp is proprietary software, we cannot examine the code and find out whether these conclusions are true. What can be concluded for sure is that there is a technical possibility of Facebook developers collecting, storing and using any WhatsApp user data, including messages.

b) **End2end encryption is not necessarily applied to all user interactions.**

For example, in the Telegram messenger, end2end encryption is applied only to specially created secret chats, while other interactions stay unencrypted (only client-server encryption is applied), and the Windows version does not have a secret chat feature at all. Most users simply do not investigate these peculiarities and use the app in the default mode. Since all user data, including transferred messages, is stored permanently on servers, nothing prevents developers from using it at their discretion.

Other messengers persistently suggest creating a backup of all data, which is then stored in the cloud in an unencrypted form.

c) **Even with end2end encryption, a lot of metadata still can be stored on servers.**

Analysis of this data can provide a lot of information about users: much more, in fact, than meets the eye. There is an interesting article on the subject by Kurt Opsahl in which he provides some examples of why metadata is really something that we should take seriously:

«They know you rang a phone sex service at 2:24 am and spoke for 18 minutes. But they don't know what you talked about.»

«They know you called the suicide prevention hotline from the Golden Gate Bridge. But the topic of the call remains a secret.»

«They know you spoke with an HIV testing service, then your doctor, then your health insurance company in the same hour. But they don't know what was discussed.»

«They know you received a call from the local NRA office while it was having a campaign against gun legislation, and then called your senators and congressional representatives immediately after. But the content of those calls remains safe from government intrusion.»

«They know you called a gynecologist, spoke for a half hour, and then called the local Planned Parenthood's number later that day. But nobody knows what you spoke about»...

The problem of metadata collection and analysis is also recognized on the highest levels. Here is an extract from the report of the UN High Commissioner for Human Rights (A/HRC/27/37):

«The aggregation of information commonly referred to as "metadata" may give an insight into an individual's behavior, social relationships, private preferences and identity that go beyond even that conveyed by accessing the content of a private communication. As the European Union Court of Justice recently observed, communications metadata "taken as a whole may allow very precise conclusions to be drawn concerning the private lives of the

persons whose data has been retained." Recognition of this evolution has prompted initiatives to reform existing policies and practices to ensure stronger protection of privacy.»

The first two issues can be solved by making an actually fully free and open source software. Some developers are already following this path (for example, the Signal messenger), so even though the most popular solutions on the market do not fit this requirement, we cannot say that these issues are completely unresolved.

The metadata issue, however, is a problem inherent in any centralized solution. The only way to solve it is to use peer-to-peer architecture.

Complete P2P design offers several benefits to a messaging service:

- Client and server functions are merged in one app, making it easier to audit the code and ensure that there are no backdoors.
- Users do not have to trust any particular party, since the whole system is run by users themselves.
- The developers have no extended access to system operations or user data; therefore, there is no sense in applying pressure on the developers by any parties who wish to influence the system or obtain user data.
- Using an appropriate system of relays, it is possible to hide user interactions, making the app relatively anonymous and secure.

This means that if we want to take the next major step in instant messaging evolution, we should develop a decentralized messenger working via a peer-to-peer protocol with an open source code.

To date, there is no truly fast, stable and convenient P2P messenger. All products are currently in a very rudimentary state and have unclear prospects, since there are some issues that are very hard to solve:

a) **There are numerous popular solutions available and the market has long been formed.**

If we come up with just another messenger, how do we motivate people to download it if it is not used by a large number of their friends? It is quite hard to promote a new product in such a crowded market even though we can offer several major improvements. To obtain a large userbase, we need both to conduct a serious marketing campaign and to introduce some killer features capable of attracting the attention of a major audience.

By merging a messenger with a cryptocurrency, we can introduce several new stunning features which were not available to developers earlier, thus greatly improving its marketing potential.

For a more detailed description of our marketing strategy and Liberdyne's killer features, please refer to the Marketing chapter.

b) **People are not willing to share their resources for free.**

During a system's infancy stage, P2P messengers do not consume any noticeable amount of resources, so most users simply pay no attention to this matter. With the growth of the userbase and the increasing percentage of mobile clients, however, full nodes will begin to consume more resources, and at some point users who are not willing to constantly share their bandwidth for free will start shutting down their full nodes, which in turn will shift the load to the remaining users, thereby increasing their overhead. This will lead to a chain reaction that creates the risk of bringing the system back to something resembling client-server architecture, where developers' servers will be the only full nodes left.

To overcome this stage, we need to offer users incentives to keep running full nodes, and that is when a cryptocurrency comes into play. Since we are merging two technologies in one product, a full node of the messenger is at the same time a full node of the blockchain. This

allows rewards to be issued for supporting both components of the system, thus completely solving the problem of user motivation.

These statements already show why distributing a cryptocurrency via a messaging platform is such a great idea, but we have only mentioned several ways that blockchain can benefit a messenger, while it actually works both ways – a messenger can also provide a cryptocurrency with some great opportunities that were not available before. Such features will be covered further in this white paper.

It is worth mentioning that a number of developers (some of whom will be discussed in the section on competitors) have already tried to develop a messenger based on a cryptocurrency platform, but to date, in our opinion, no one has been able to present a complete, high-quality product. One of the main problems of the unsuccessful attempts is that messengers are built on the basis of already-working blockchain platforms, which are not optimized for such applications. To create an adequate product, both components must be developed in coordination.

III. PoW vs PoS

1. Brief description of the problem

Bitcoin was not technically the first digital currency, but it was the first project to solve one last major problem on the way to building a decentralized payment system – determining the order of transactions without the involvement of a trusted intermediary.

In centralized systems, such an order is determined by an appointed party, who simply places the timestamp the moment it receives a transaction. In peer systems, transactions may come to different peers in a different order, so there is a need to come to an agreement (consensus) via some kind of a voting mechanism.

The simplest implementation of such a voting mechanism is the one-entity-one-vote principle, but unfortunately, in anonymous and pseudonymous permissionless systems, one entity may create multiple representations and perform a Sybil attack. Therefore, a Sybil-proof peer consensus solution had to be developed.

The Nakamoto consensus, introduced in the Bitcoin white paper, relied on executing a certain amount of work as proof of any given party's right to add a block of transactions to the database. All other system participants should accept a block only with attached proof of execution of the determined amount of work. One of the key features of the proposed approach is the impossibility to falsify such proofs. If someone tries to alter the transaction history, he or she will have to redo all the work from the genesis block (or any other desired branching point) to the current block of the system.

The proof-of-stake algorithm introduced later uses a similar concept, but instead of executing work, the right to produce a block is based on the number of tokens in one's possession. The fundamental difference of this proposed solution is that proofs are based on an intrinsic resource, whereas PoW relies on an external resource.

Despite the fact that PoS allowed developers to build faster systems and greatly reduced the amount of consumed resources, it is not considered the perfect substitution for PoW due to several problems arising from the mentioned difference.

The main issue is PoS systems' vulnerability to so-called long-range attacks, to which PoW is considered resistant. We will not describe these problems in detail, since a great deal of information on the subject can be found over the web. We can only state that, to date, PoW systems (such as Bitcoin) are still considered safer, and PoS systems' reliability is put to doubt.

We do not share this opinion, and we will show that PoW protocols are not significantly more secure than PoS and that, considering all the benefits we can get from moving to PoS, we should finally leave PoW systems behind.

2. Objectivity or subjectivity?

The Nakamoto consensus protocol and other PoW-based protocols are considered objective because any newcomer may establish the valid current state of the blockchain knowing only the protocol rules and given the fact that at least one node has provided the correct chain.

Since PoS is vulnerable to the rewriting of history, a new node may not be able to distinguish the number of resources contained in the adversary's altered version of the blockchain from the number of resources in the main chain, because they can both formally satisfy the protocol rules and have the same weight in the system from the point of view of an external observer. Unlike computations, stakes can be replicated on any arbitrary forks at a near-zero cost. That is why, in the case of PoS protocols, we have to trust the source (or multiple sources) that provides us with the current state of the blockchain, and that is why PoS protocols are referred to as subjective.

PoWs' objectivity, on the other hand, is based on the lack of need to trust any third parties to verify proofs and establish which chain contains more computations and therefore is the right one. The problem is that, while as an abstract model it really works this way, when it comes to real-life implementation, it instantly becomes much less objective than it seems to be.

In Bitcoin protocol, the amount of work produced over a block is proven by the hash of the block, which should be below a given target. To verify the validity of the blockchain, a node must calculate a SHA-256 hash for each block.

One can try to accomplish this task with a pencil and paper. According to the research, the rate may reach 0.67 blocks per day, which means it would take only about 2,300 people constantly calculating hashes for a year to verify the Bitcoin blockchain as of the beginning of 2019. Another option is to take a more pragmatic approach and use software to accomplish the task in a few minutes, but that is the point where objectivity starts to fade away, because this means that we have to trust the developers of this software.

The problem is that, in practice, blockchain not only operates on the protocol layer but also involves the client layer, the OS layer etc. The objectivity of the protocol cannot be transferred to these layers; thus, overall objectivity is in practice unattainable.

If we try to evaluate whether there is any difference in the amount of trust involved in downloading software from a third party and downloading a blockchain state from a third party, we may conclude that there is none, actually.

The overwhelming majority of new Bitcoin users start using the system by downloading a client from a well-known developer (e.g. Bitcoin Core). Any client can be easily compromised to recognize faulty hashes as valid ones, thus allowing the developer to make all nodes that use the given client switch to the developers' corrupted blockchain from the valid one any time he or she wishes (or just steal users' private keys, if the malicious developer prefers a simpler approach). The only way to make sure the client is secure and the verified blockchain objectively contains the stated amount of work is to conduct a full audit of the source code (including every update) or code one's own client, which is done in close to zero cases.

When a new user joins a PoS system, he does exactly the same, except that PoS software will not be always capable of distinguishing the correct system state reliably if different versions of the blockchain are provided by nodes. This is why the current state is usually provided by the software client developer along with the genesis block (or a set of trusted nodes is predefined in the preferences for this purpose). It involves about the same amount of trust as in the PoW's case.

This means that the PoW's objectivity is mostly an abstract model, which can be used for academic purposes but cannot be properly used in real-life scenarios.

If we take a fully paranoid approach, we may come to the conclusion that, to verify the objective proofs of work contained in a newly downloaded Bitcoin blockchain, we have to trust:

- The person who developed the Bitcoin software client, because he could have compromised it;
- The administrator of the resource from which we downloaded the client, because he could have replaced the legitimate client with a compromised one (this issue can be solved with a cryptographic signature, but then we would have to trust the certification authority);
- The operating system developer, because he could have compromised the OS;
- The device vendor, because he could have preinstalled compromised software; and
- The hardware manufacturers, because they could have flashed compromised firmware.

It seems that to take full advantage of PoW's objectivity and verify the proofs contained in a newly downloaded Bitcoin blockchain without actually having to trust third parties, we would have to build our own device, starting from the CPU and all other components, and code our own firmware, operating system and Bitcoin client. In addition, we should code everything in a low-level language, because any higher-level programming language translator could have been compromised as well.

For this reason, we consider Bitcoin's objectivity to be nothing more than a spherical cow in a vacuum – an abstract model with no relationship to the practical environment.

3. PoW's practical security level

Besides the problem of subjectivity, PoS algorithms are also sometimes considered less secure from the point of view of the amount of resources needed for a successful double-spend attack.

In reality, the level of security highly depends on the issuance rate, for it defines the size of the reward for block creation, and besides, the security model of PoW blockchains scales poorly.

It is well known that to successfully attack a PoW system one will need >50% of the total mining power. Although there can be certain conditions that will allow one to succeed with many fewer resources, such a scenario is rather complicated and not guaranteed to be successful, as most major mining pools may refuse to mine on top of an adversary's chain, even in the case that the adversary succeeds in building a longer chain for the required time period.

Indeed, 50% of the current Bitcoin hashrate, which is measured in exahashes, seems an enormous number. This leads PoW proponents to the conclusion that Bitcoin's architecture still seems to be the most robust.

To verify this statement and see whether it is actually so difficult to execute a double-spend attack on Bitcoin, we should evaluate the actual system's attack resistance in as close to a real-life attack scenario as possible.

A 51% attack can be carried out in different ways:

a) **Controlling the required computing power.**

This is a more likely scenario, and for this reason it is the main point of our interest.

Assume that n is the total system token supply and x is the number of recourses needed to perform a successful attack, measured in tokens at their current market exchange rate. Then $x \div n = k$, where k is the coefficient of the system's practical attack resilience.

Since Bitcoin's attack resistance is based on an external resource, the coefficient will not be static and will constantly change over time, depending on multiple conditions (i.e. the BTC market value and the hashrate of the system). Due to the preset issuance rate, however, the coefficient should hold at an approximately same scale and reduce after each halving.

Let us pick some remarkable date to count the coefficient. The moment of Bitcoin's peaking in value on Dec 17, 2017 looks an appropriate option, for Bitcoin's market rate exceeded $20,000 that day, and we can presume that it may have been one of the most opportune moments to execute an attack to date.

On Dec 17, 2017, the hashrate of Bitcoin reached about 14,630,524 TH/s, which means that possessing about

$$14{,}630{,}524 \div 2 = 7{,}315{,}263 \text{ TH/s}$$

would have allowed one to execute a successful attack. One of the most popular and efficient ASICs for Bitcoin mining at that time was Antminer S9, which produced about 14 TH/s at a cost of $2000 in retail. So, to execute a successful attack we would have needed

$$7{,}315{,}263 \div 14 \ = 522{,}519 \text{ Antminers}$$

which could have cost us

$$522{,}519 \times 2000 = 1{,}045{,}038{,}000 \text{ USD}$$

Considering that we know the Bitcoin total market cap as of that day, we can finally calculate the Bitcoin attack resilience coefficient:

$$1{,}045{,}038{,}000 \div 336{,}433{,}998{,}575 = 0.003$$

This means that we needed to invest only 0.3% of the total Bitcoin circulating supply to perform a successful attack, and it does not seem to be as secure as PoW supporters state.

Furthermore, with the growth of the market cap, we are going to need more and more hardware to hold the coefficient, which is why the model scales poorly. If we assume that Bitcoin became the world currency and its real value grew to the extent of USD (which is roughly about $20 trillion), then to maintain the same coefficient we need to engage about $600 billion worth ASICs, which will presumably consume petawatt-hours of energy.

b) Bribing miners for a temporary attack.

Instead of controlling the required hashrate directly, the adversary could bribe major miners or mining pools to secretly mine the adversary's chain for a required period of time. Instead of the security coefficient derived in the previous clause, we should consider the price of the attack per time.

We suppose this vector of attack in practice improbable to occur in the main fork for social and economic reasons. If we assume that major miners could coordinate their actions to commit an attack, they would prefer to gain direct profit instead of acting in favor of the adversary. Moreover, if miners committed such an attack, it would undermine the credibility of the entire system and cost miners severe potential losses from the subsequent market value drop. Under such circumstances, the adversary would need to offer a very large reward, which could exceed his or her potential gains.

Given the results above, we can conclude that the alleged PoW's advantages do not in practice provide a significant advantage over PoS. The problems of subjectivity and insufficient security are exaggerated, and there are no reasonable obstacles to a full transition to PoS algorithms.

Implementing a set of specific features can help build a PoS algorithm with a practical security level that is at least not inferior to classic PoW implementations or that even surpasses them, as will be shown later in this white paper. Besides, PoS systems, featuring an endogenous security model contrary to exogenous PoW security, scale much better and have no problems with any potential growth in value.

IV. Dynemix Cryptocurrency Platform

1. Brief overview

Dynemix is a decentralized, permissionless, account-based blockchain system powered by a unique Proof-of-Stake BFT consensus protocol.

Dynemix is designed to achieve one goal – to become the first widely adopted decentralized means of payment (commonly called a cryptocurrency) and compete with conventional centralized payment infrastructure on equal footing.

Dynemix operates in a symbiotic relationship with the Liberdyne messenger, which allows it to introduce features not previously available and to progress toward the stated aim.

To achieve the declared goal, Dynemix is endowed with the following features:

a) **Transactions are free.**

Unlike other blockchain systems, which feature market-determined fees for all transactions, common transactions in Dynemix are free. Fees are charged only for business-related transactions that require multi-outputs.

b) **All transactions are finalized within 6 to 16 seconds after being sent to the system.**

Unlike other blockchain systems, which allow transactions to be added to the block at the discretion of the block producer, every transaction broadcast into the Dynemix network is added to the next block, which is instantly finalized. There is no need to wait for additional confirmations.

c) **Dynemix scales to 10,000 TPS and higher.**

Our sharding solution allows Dynemix to scale up as much as needed. The system can scale to 10,000 TPS and beyond with moderate hardware and bandwidth requirements.

d) **Dynemix is highly decentralized.**

Unlike many recent blockchain projects, which solve the scalability issue by adopting highly centralized architecture, Dynemix is designed specifically to provide the highest level of decentralization, which surpasses even the level of classic PoW blockchains, such as Bitcoin.

e) **Dynemix is highly secure.**

With the help of a two-layer security model, Dynemix manages to increase scalability without the need to sacrifice security.

f) **Dynemix can be minted on a common PC at home.**

Due to its excellent optimization, the use of sharding technology and the new consensus protocol, it is possible to run a full node on common home-class hardware. No professional hardware is required.

g) **Dynemix introduces a new economic model.**

The system features a unique coin issue and reward distribution system that provides a solution to the problem of economic development.

h) **Dynemix is compatible with financial privacy.**

Dynemix is designed to use a new cryptographic solution that encrypts transaction data and can allow financial privacy to be provided to all users.

i) **Dynemix solves the entry-threshold issue by being integrated into Liberdyne.**

Dynemix is integrated into the Liberdyne messenger to provide users with a more familiar experience. Newcomers do not have to study the peculiarities of cryptocurrency technology to start using Dynemix. Instead, the user only needs to install Liberdyne and start using a familiar type of app that happens to have additional functions, which are available through a simple and intuitive interface. Minting functions are set up automatically, so the user does not need to do anything at all.

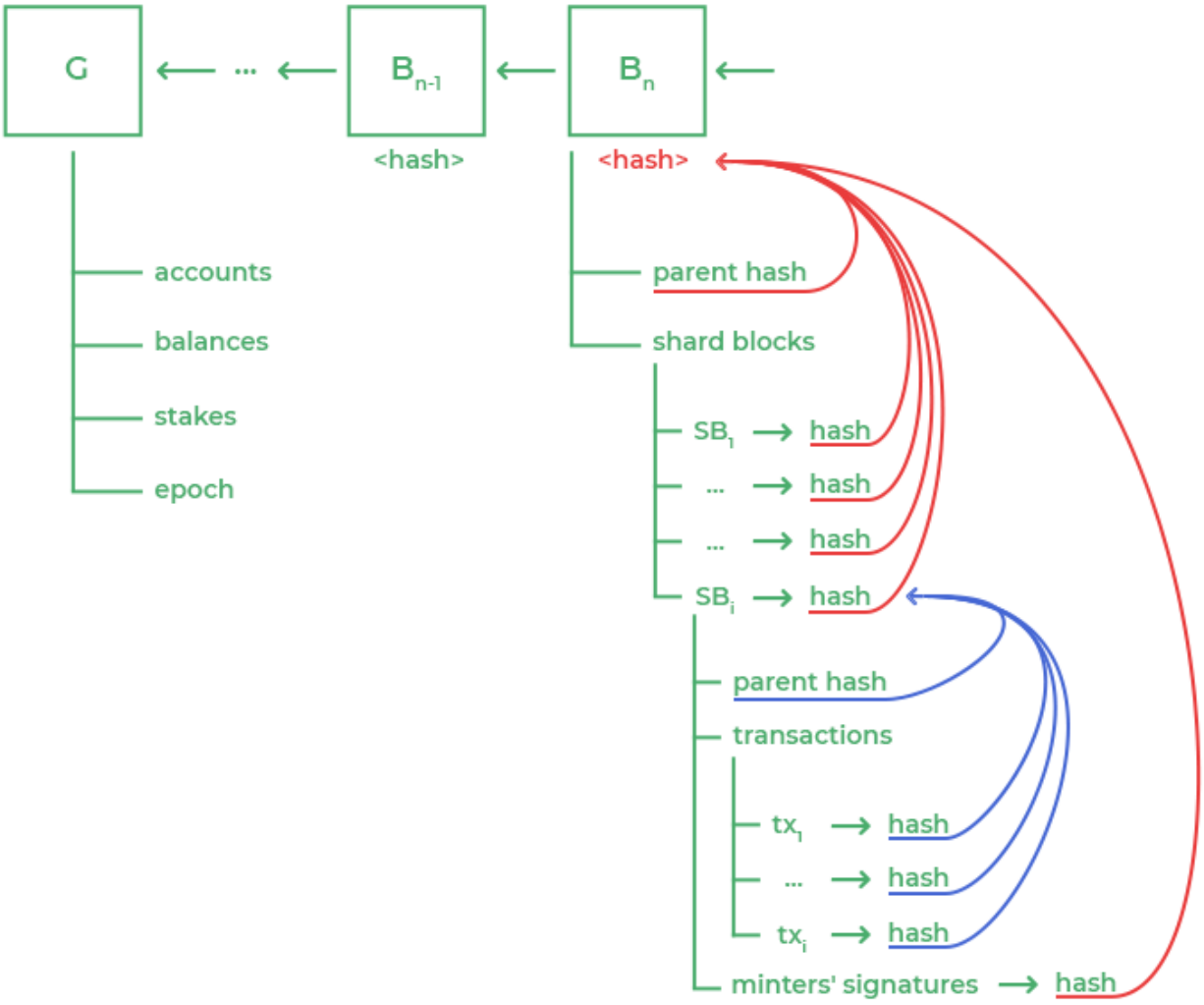
The combination of these features makes Dynemix the ultimate blockchain payment platform and a true breakthrough in cryptocurrency technology.

2. Data structure of Dynemix

Dynemix is based on an ever-growing transaction ledger called a blockchain. Dynemix blockchain consists of a sequence of master-blocks that are added in linear order.

Unlike most other blockchain systems, participants in Dynemix can operate not only directly with complete data blocks but can also employ a simplified data structure, which is derived from blocks and called *a quilt*.

1) Blockchain – master structure



Essentially, each master-block represents a collection of shard-blocks and contains the following data:

- the hash of the previous master-block;
- a set of shard-blocks;
- a set of minters' signatures for shard-blocks.

Each shard-block contains:

- the hash of the previous master-block;
- a set of transactions.

All data is organized into a hash tree.

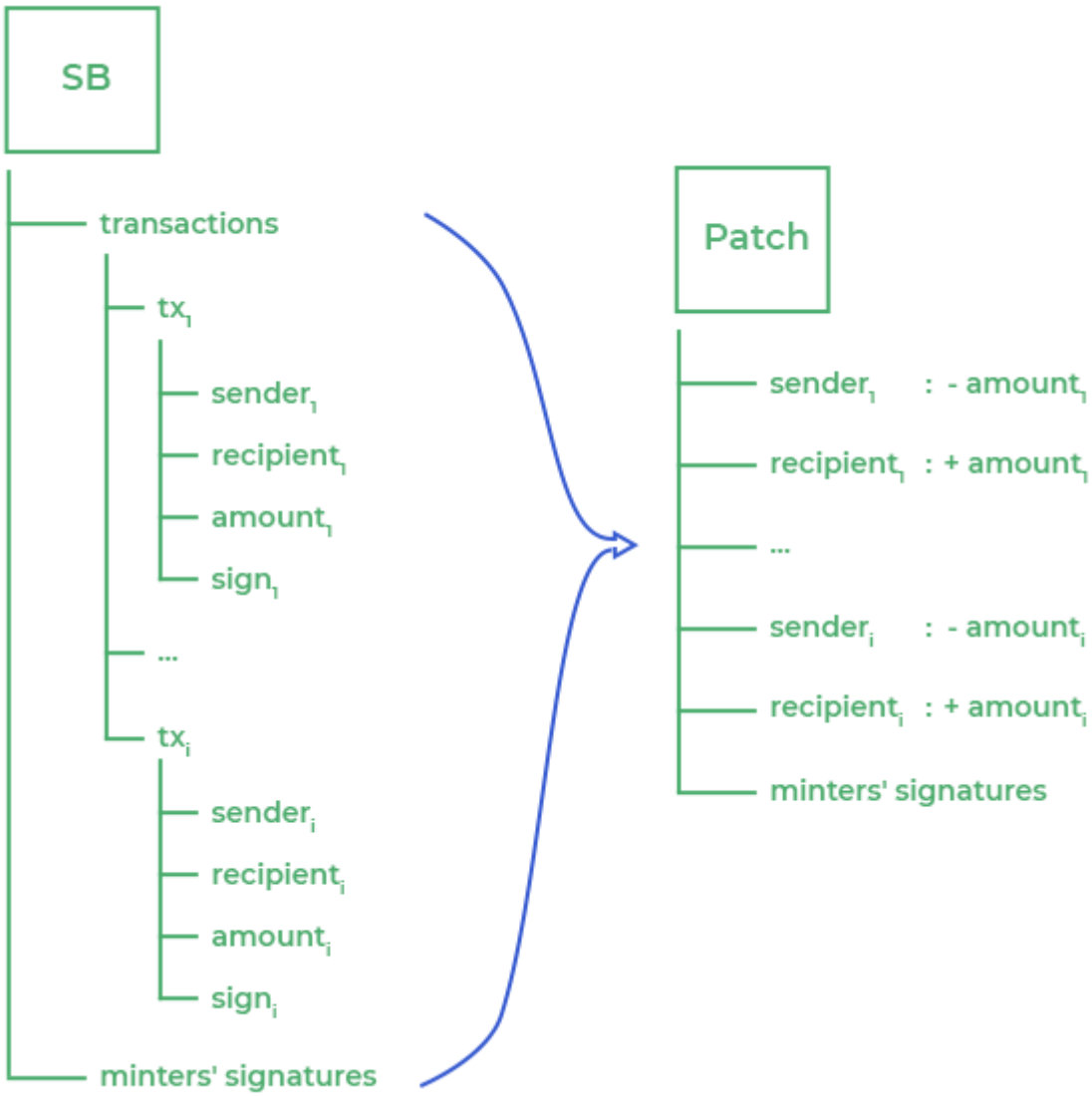
2) Quilt – derived structure

Quilt is designed to provide better optimization and scalability. Operating in the quilt mode is most efficient in the case of balances' homomorphic encryption implementation, which is the particular setting for which quilt was designed. Even without encryption, however, quilt can provide an increase in performance, although the result is not as impressive as in the specified case.

In the early stages, quilt will not be engaged, and Dynemix will fully operate in the blockchain mode, which assumes the propagation of complete data blocks among all nodes.

At a certain point, however, balance encryption implementation should be considered in order to comply with financial privacy standards. The system was initially designed to provide a smooth subsequent transition to an encryption scheme.

Balance encryption will inevitably affect performance. The weight of transactions and computational overhead will dramatically increase, thereby possibly dropping throughput on the blockchain layer by an order of magnitude. To partially compensate for the performance decrease while maintaining the desired level of scalability and decentralization, full nodes will be allowed to operate with a pruned data structure that is derived from the underlying blockchain.



Once minters of a shard construct a valid shard-block, they derive a data structure called a *patch* from the newly constructed block. While the shard-block contains transactions with all attributes included, the patch contains only changes of the state that are caused by transactions. The hash of the patch is included in the hash tree of the shard-block.

At the master consensus stage, instead of exchanging complete shard-blocks, minters exchange patches and partial hash trees of shard-blocks, along with signatures. Thus, after reaching a master consensus, each minter will now possess a complete set of patches, which form a quilt. Essentially, a quilt represents the set of changes applied to the system state by a master-block of a given height.

We assume that by the start of each round all minters will possess the current state of the system. Upon applying a quilt, minters will obtain a new state. Operating with quilts allows a significant decrease in computation and communication overhead that occurs after the shard consensus subslot.

It can be easily observed that once full nodes begin operating in the quilt mode, the system will become less secure, as the validity of the state change is not verified beyond a shard consensus. The inevitable loss in security is circumvented by the presence of authorized master-nodes that perform fishermen functions.

For a more detailed description of the overall security model, please refer to the next chapter.

Although Dynemix will initially be launched without quilts, the following description of the system accounts quilt's implementation as the final design that the system should obtain. Please note that some of the described functions of the system will not initially work completely in accordance with the description below.

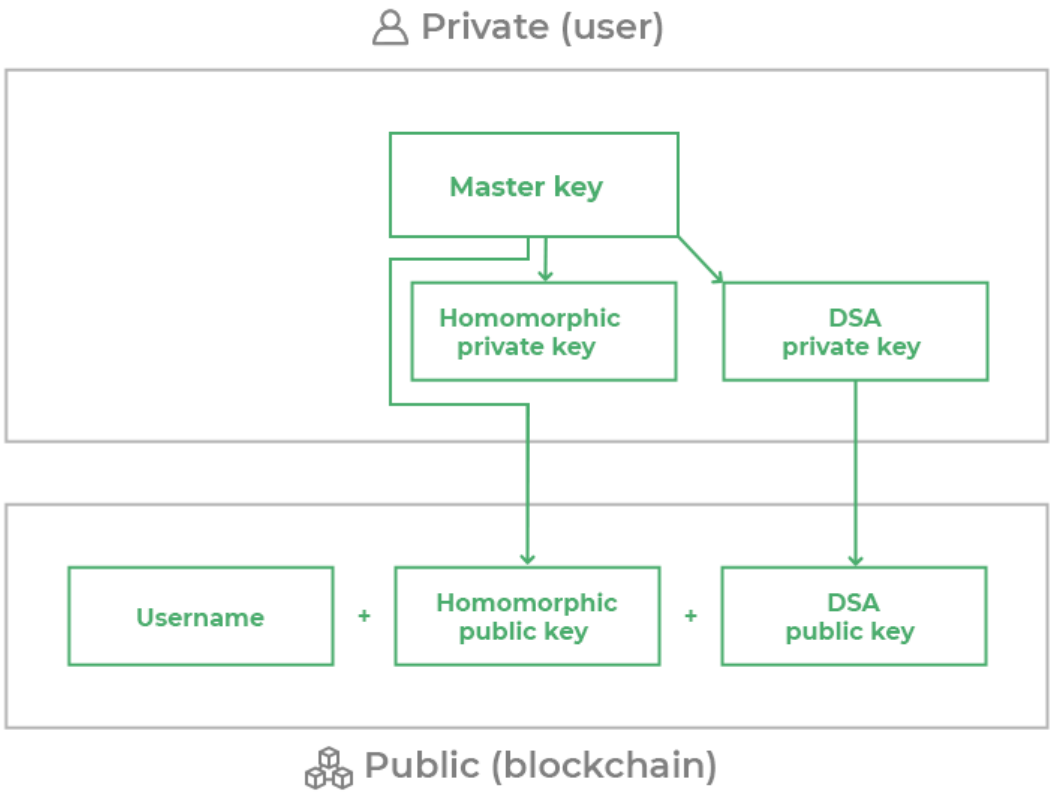
3. Accounts in Dynemix

1) Dynemix Namespace

A user can interact with the Dynemix system on two levels:

- **With the help of public/private key pairs.** This level resembles the approach used in most blockchain systems. Capabilities on this level are limited to issuing transactions.
- **With the help of a registered account.** Most actions within the Dynemix/Liberdyne system require registration.

During the registration process, not only are key pairs generated, but a username is also attached to the public keys. The entirety of all unique usernames chosen by the users during a registration forms the Dynemix namespace.



Registration is a special type of a transaction. Once a user registers, it is impossible to alter or delete the username from the blockchain. That prevents any censorship attempts in the namespace service.

Registration is implemented for the following purposes:

a) **Improving user experience**

The username database is stored within the blockchain, which guarantees the possibility of resolving a username to the public key at any time. Hence, any interactions within the system (for example, sending tokens) can be performed via a username. This makes the system simpler and easier for users to understand.

The namespace can be also used by other projects within the Dynemix ecosystem or even third-party centralized services. For example, by being combined with integrated decentralized cloud storage, the namespace can allow users to create different types of content that can be accessed by various services, apps or other users via the username.

b) **Statistics generation**

Since the system is completely decentralized, it is difficult to determine even the approximate number of users. The number of public keys with non-zero balances may help, but users can easily generate an infinite number of one-time keys and split their tokens among them.

Registering multiple accounts does not make much sense, since users could still employ different public keys as separate wallets but have only one registered account at a time.

c) **Spam protection**

This feature is mostly useful for the Liberdyne messenger, as well as other potential projects based on the Dynemix platform, but it can be also used to counter the transaction spam issue, if such occurs.

Initially, we plan to support free registration without any restrictions (except for CAPTCHA protection from bot registrations). In the case that we face significant trouble with spam and undesired content distribution, it is possible to complicate the registration process to make the Sybil strategy for malicious activities more resource-intensive.

2) Verified namespace

Since Dynemix is a decentralized system and the process of registration is decentralized as well, it is impossible to use standard methods of cyber-squatting resistance.

To solve the problem of cyber-squatting it is reasonable to add a centralized service – the verified namespace.

The verified namespace works similarly to verified accounts in centralized social networks (e.g. Instagram and Facebook). Once a publicly known user wants to get a confirmation mark indicating that a particular account belongs to him, he will provide the requested information and get a verified username. These usernames will form a separate namespace and will have priority over identical usernames from the general namespace in some scenarios.

Verification can be also used to add a legal provable binding between a Dynemix account and a certain entity (a sort of certification), thus allowing entities that wish to use the platform for business purposes to gain more trust from users.

Another critical feature of the service is its ability to restore an account in case of key loss. Account names may be valuable in themselves as a marketing tool, especially for large companies that make significant investments in brand marketing. The risk of irretrievable loss of the account may discourage such companies from using all of the system's capabilities. Centralized account delegation may solve this problem and provide the necessary enforcement of the ownership of the account.

4. Dynemix units – dynes

Dynemix units are called dynes. Dyne is a derived unit of force in the CGS system of units. Currently, the term is barely used, so we will likely be able to give the word a second life and at the same time avoid confusion.

We like the word for its association with power and, besides, it just sounds cool.

Dynemix is not designed to support Turing-complete smart contracts, and hence tokenization will not be supported either. Dynes will be the only units circulating within the Dynemix ecosystem.

Adding certain on-chain derivatives may be considered, however, if the platform's economy requires it during the evolution process. It is difficult to foresee the path of development at this initial stage.

Dynes are utility tokens, which bear no other function except being a universal medium of exchange and are not backed by any real assets or obligations. Within the platform, dynes can be also used as stakes in the minting process.

The initial issued volume will reach 1,000,000,000,000, or 10^{12}, dynes.

The minimum fractional unit is equal to 0.000001, or 10^{-6}, dynes.

Starting from the second master-block of the blockchain, new dynes will be issued as rewards for system support with each new master-block.

For a more detailed description of the dyne issuance model, please refer to the Economics chapter.

5. Dynemix transactions

1) Transaction types

Account transactions	
Register	Registers a new user account
Register verified	Registers a new verified account
Recover	Allows public keys to be reset with the help of the master key (changing the password or master key)
Token transfer transaction	
Transfer	Transfers tokens between public keys

Service transactions	
Stake/unstake	Stakes or unstakes a specified number of tokens for minting
IMIN	Indicates readiness to participate in the minting of the next block
Proof of fraud	Incriminates a minter in an attempt at fraud, slashes the minter's stake and restricts him from further participation
Administrative transactions	
Administrative	Allows special administrative measures to be applied in the case of their adoption

2) Spam protection

Since we follow the concept of free transactions, it raises the problem of spam – instead of sending one transaction with the desired amount (say, 100 dynes), a malicious user could deliberately send 100 transactions of one dyne each, hence creating an unreasonable load on the system. To solve this issue, we adopt several restrictive measures:

a) **Limiting the number of transactions**

The system will accept only one transaction from each address in each block. This limitation will not significantly diminish the user experience, since Dynemix has a 10-second block time, and in very few cases will a user need to make transactions faster than that. If the user needs

to make several consecutive transactions, he or she may send them to the network and wait until they are processed.

Business scenarios often require multiple transactions to be processed from one account at a time, however. To meet the needs of business accounts, Dynemix supports multi-output transactions. To protect the system from spam, fees are charged for each output of such transactions. Unlike most blockchain systems, which feature market-defined commissions, fees in Dynemix are predefined to improve the user experience.

b) **Limiting the transaction amount**

Another antispam measure is limiting the minimum amount of a user-generated transaction to one dyne. In case the total market cap grows high enough, this limit can be lowered to meet user needs.

In case these measures prove insufficient, additional restrictions can be applied:

a) **Limiting the daily amount of free transactions**

According to the gathered statistics, it is possible to estimate the daily average number of transactions and limit free transactions to a certain number that would satisfy the needs of most users.

b) **Charging monthly fees**

We can set monthly fees for using the system. When a user issues his or her first transaction in a month, it should contain a predefined service fee, which is obtained by minters. After paying the fee once, the user can freely issue any number of transactions until the prepaid period expires. This approach resembles the conventional banking experience, as users usually pay annual or monthly card issuance and service fees to banks, which is why this approach should be familiar to most users.

6. Financial privacy

Most cryptocurrencies disclose transaction data, i.e. the number of tokens sent, the sender and the recipient. Since we are creating a platform for everyday payments, it would be advisable to solve this issue and ensure compliance with banking secrecy. A system that allows anyone to see the current balance and the entire payment history of any user is obviously less attractive to consumers than a system that can assure their financial privacy.

When we started the project, several platforms (Monero, Zcash, Dash etc.) offered solutions that allowed the amount and/or the sender and/or the recipient of a transaction to be hidden.

The problem was that all those platforms were UTXO-based blockchains, while Dynemix was meant to become an account-based system (a UTXO model requires higher traffic and storage overhead, which is intolerable for a system that is meant to scale up to thousands of TPS). Besides, the existing solutions drastically increased the transaction weight, RAM use or computation time.

No balance-hiding solution for account-based blockchains existed at the time, which is why a completely new approach was required.

Financial privacy in relation to blockchain consists of two conditions, each of which requires a separate technical solution.

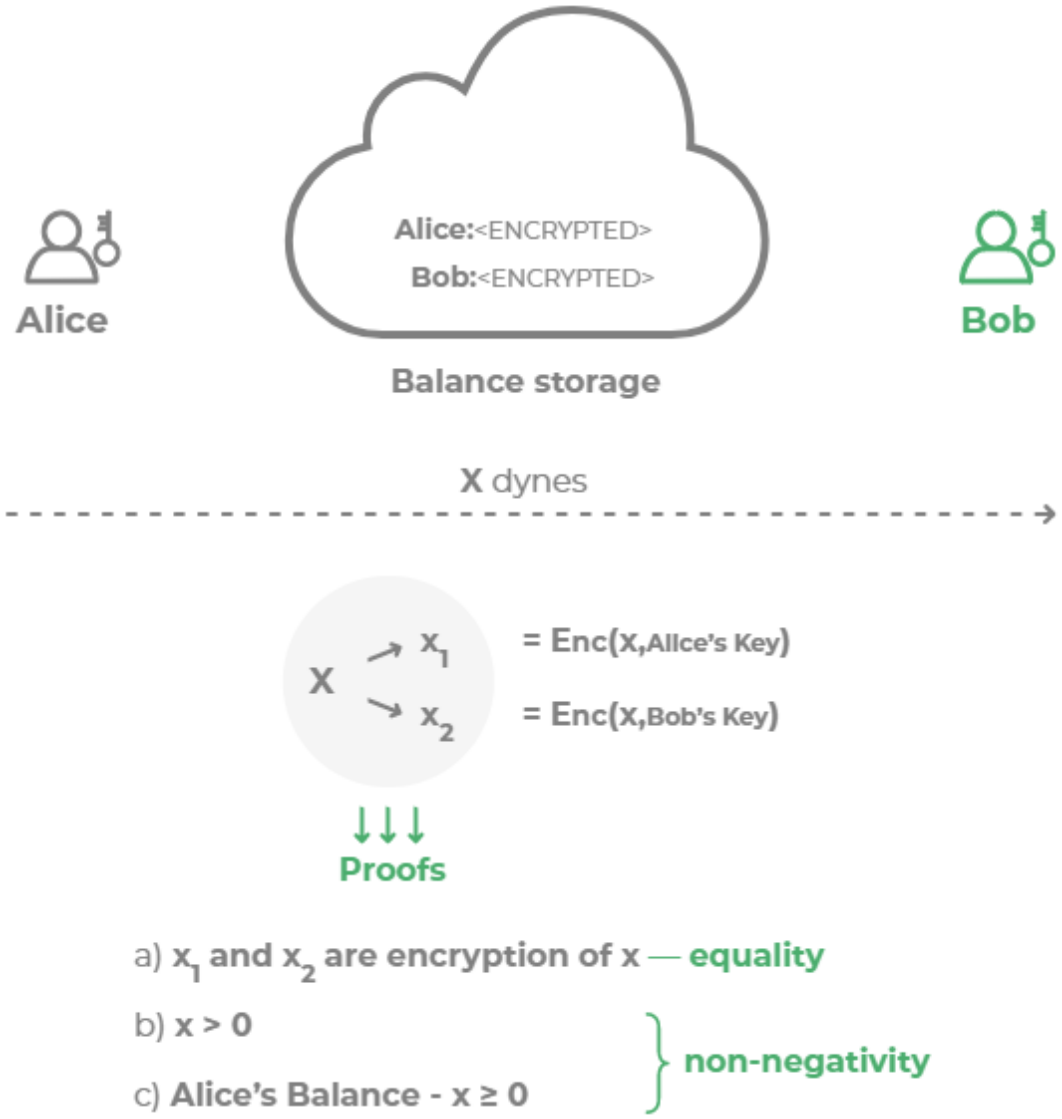
1) Hiding transactions' balances

We believe that this condition is of the utmost importance – hiding transactions' amounts allows for the hiding of account balances, which are sensitive information that the majority of users would not want to be disclosed.

To solve this issue in an account-based system, we can use homomorphic encryption, as it allows computations on the ciphertext, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext.

Instead of storing balances and transaction amounts as plaintexts, we can store encryption keys and encrypted balances.

Assume Alice wants to send x dynes to Bob. She also wants her balance and the transaction amount to be hidden from Eve. As our blockchain is public, the only way to achieve this is to introduce encryption.



Instead of subtracting x from A and adding x to B, roughly speaking, we subtract encrypted x from encrypted A and add encrypted x to encrypted B.

Of course, Alice and Bob have different encryption/decryption keys, which is why x should be encrypted twice (with Alice's key and with Bob's key) to be able to be subtracted from Alice's balance and added to Bob's balance.

Then the following problem arises – Alice should prove that:

- both ciphertexts encode the same value;
- this value is positive;
- her new balance is at least non-negative.

To achieve this, we need (non-interactive) zero-knowledge equality and range proofs.

After a minter receives the transaction, he or she checks the validity of the attached proofs and performs the subtraction of x_A from Alice's balance and the addition of x_B to Bob's balance.

If Alice wishes to perform a transaction with the disclosed amount, she can easily do so. This may be necessary if Alice wishes to leave evidence of a transfer of a certain amount to Bob in the blockchain.

2) Hiding recipients' IDs

This condition is also important, because disclosing the recipient of the transaction allows for the identification of connections between users, but we suppose that this is not as crucial for most consumers as is balance hiding.

Since Dynemix supports multi-output transactions and given homomorphic encryption implementation, the solution to this issue is trivial – instead of sending a transaction with one output (Bob's account), Alice sends a transaction with multiple outputs, one of which is Bob's account and the others are randomly chosen registered accounts. The amount transferred to random accounts is equal to zero, while the amount transferred to Bob is the actual desired sum.

From Eve's point of view, Alice sends an unknown number of coins to multiple recipients, and Eve cannot conclude that either of them are real recipients rather than just random extras, nor how many coins each of them received.

Alice can arbitrarily choose the number of extra outputs, assuming that the more outputs she includes, the more she confuses Eve, but at the same the more she must pay (as fees are charged by minters for each additional output's processing).

Alice can also issue multiple transactions with multiple outputs, thus hiding the mere moment of the actual transaction's issuance. The degree of obfuscation depends only on the number of dynes Alice is ready to spend as fees for extra outputs.

A significant advantage of this approach is its high customizability on the client level. Since no patterns are predefined by the protocol, it complicates analysis attempts for a potential adversary.

Interaction obfuscation turns out to be a paid feature, but it is justified by the excessive load put on the system by such transactions. We suppose that balance encryption will be sufficient for most users and that multi-outputs will be used infrequently.

3) Problems of proposed solution.

Before homomorphic encryption can be implemented, we must solve several major problems that arise with the proposed solution.

a) **Constructing suitable ZK-proofs**

Although homomorphic encryption theoretically can allow our standards to be met, as for a single account only an encrypted balance and public keys must be stored on-chain, thus saving a lot of space, the problem lies in the construction of the required ZK-proofs.

The only currently available working solution is an additive modification of the El-Gamal encryption scheme, which requires calculating a discrete log in order to decrypt a balance. This task is NP-intermediate, and thus the solution is completely impractical. At the same time, constructing ZK-proofs for much more convenient cryptosystems (e.g. Paillier) could require a lot of time and resources for proving and verifying, and the proofs could be much larger than expected.

We do not find it reasonable to use any solution that cannot allow a transaction to be quickly constructed even on a mobile device, which is why currently available concepts seem inappropriate. To develop a practical implementation that can meet our needs, further research is required. We are developing a system with a foundation for the quick and painless

implementation of homomorphic encryption (the architecture is initially optimized for operation with encrypted balances) when a suitable solution is fully developed. We cannot reliably claim when this may happen, however.

We can note that homomorphic encryption and ZK-proofs as technologies are still in their infancy and improved solutions have yet to be developed. We should also note that the technology is relevant not only for blockchain systems but to various spheres of the computer industry, which raises our expectations for the potential rate of progress.

For example, hardware giants are working on RAM-encryption technologies that can provide more security to cloud computing, but the currently presented solutions still feature computations with raw data on the CPU layer. Building a fully secure VM would require fully homomorphic encryption and computations directly on the encrypted data (the problem is briefly outlined here). For this reason, we can expect low-level optimizations that will help us build a more scalable system.

b) **The high risk of hidden attacks**

The encryption of all account balances obstructs the detection of attacks.

If the adversary exploits a bug that allows the creation of new coins in an arbitrary amount, it can have a devastating effect on the system. When all balances are publicly observed, it will likely be detected quickly, and countermeasures will be applied (a rollback, in the worst case). When balances are encrypted, however, it may stay unnoticed longer, which is an absolutely unacceptable outcome.

Although the state transition remains verifiable and even in the presence of master-nodes, which are expected to help keep the system secure and ensure the sufficient level of security, we suppose that it would be dangerous to implement the described technology on the platform scale until it was redundantly researched and tested.

Any possible additional measures to mitigate the stated risks should be also considered, even if they come at a cost.

7. Dynemix state transition

As stated, a cryptocurrency platform that can meet all of the requirements to compete with centralized payment systems cannot be based on any existing protocol for reasons of technical limitations. We had to design our system from scratch and apply many unique features to achieve our goal.

More details will be available in the technical documentation. Here, the protocol will be described in broad strokes, and its most important features will be explained in the next chapter.

1) Stakes

a) **Staking**

Everything starts with stakes. Any registered user can send a stake transaction with an arbitrary amount, thus indicating his or her intention to participate in block production.

As we intend to democratize block production and attain a high level of decentralization, we cannot set a high minimal stake threshold. Requirements that are too low, however, can lead to abuse and negatively affect security, which is why a point of balance should be determined.

A stake transaction's amount is publicly visible; hence, minter candidates have to partially disclose their balance to the extent of the stake amount.

b) **Restaking**

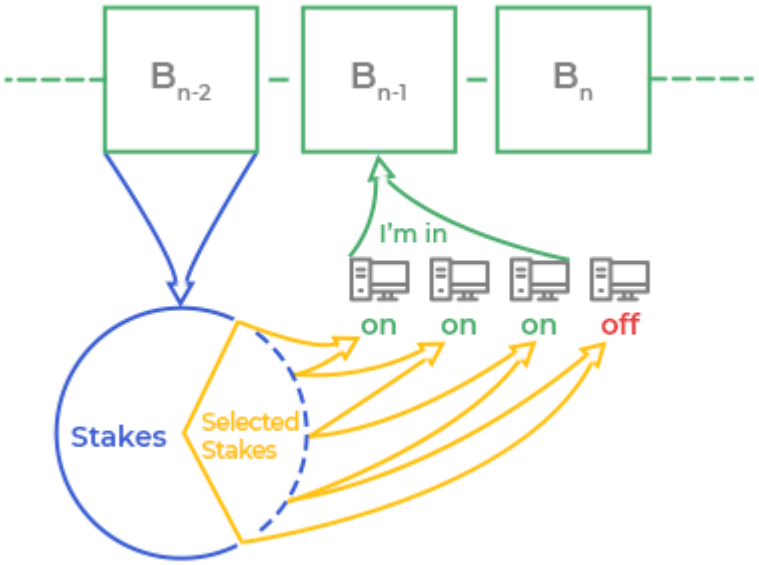
Since we initially assume that minting will be performed not only by users who adhere to a professional approach but also to a substantial extent by common users with consumer-level hardware, we cannot predict the percentage of stakeholders with 100% uptime. Although we implemented an adaptive algorithm for minter window selection that accounts for an average proportion of the available stakeholders, if the dispersion grows too high, it can have a negative effect on several properties of the system.

If we encounter such a problem, it may be solved by setting a TTL for stake transactions or suspending the stake if the stakeholder skips a block. In this case, stakeholders will need to restake intermittently to indicate their availability.

c) **Unstaking**

If the user wishes to unblock the staked dynes to be able to spend them, he or she sends an unstake transaction. The staked amount becomes spendable after two blocks are minted above the block that contains the unstake transaction, which makes Dynemix more minter-friendly than other PoS systems, as the stake in the latter is typically blocked for a long period of time for double-spend protection.

2) I'm in



Block B_{n-2}

In a pseudorandom manner with the master-block's hash as a random oracle, a set of eligible minter candidates for block B_n is selected from the many accounts with active stakes. The stake amount correlates with the probability of being selected for minting, as the weighted sampling algorithm is employed.

Block B_{n-1}

Stakeholders picked by the algorithm send special IMIN transactions to the minters of block B_{n-1} to indicate that they are ready for minting. This is required to exclude inactive stakeholders and ensure that during the minting of block B_n most of the assigned minters within each shard will be available.

IMIN transactions provide a solution to the problem of unavailable stakeholder nodes, which are considered faulty in terms of the consensus algorithm.

Since we are assuming that a substantial proportion of stakeholders will consist of ordinary users whose nodes may be unavailable much of the time, there is a serious risk that liveness may be threatened by a large number of inherently faulty nodes. This is why an algorithm that verifies availability immediately before the minters are designated is required. The solution is detailed in the next chapter.

3) Sharding

Block B_{n-2}

↓ According to the average number of transactions per block in a recent period, the optimal number of shards that the system will split into for minting block B_n is determined.

Block B_{n-1}

↓ According to IMIN transactions, a minters' committee for block B_n is formed.

Block B_n

The minters' committee is algorithmically sorted by shards and split into a set of shard committees.

Since all information required for sharding is contained in blocks B_{n-2} and B_{n-1}, the shard committee's designation for block B_n becomes consistent throughout the network after block B_{n-1} is propagated.

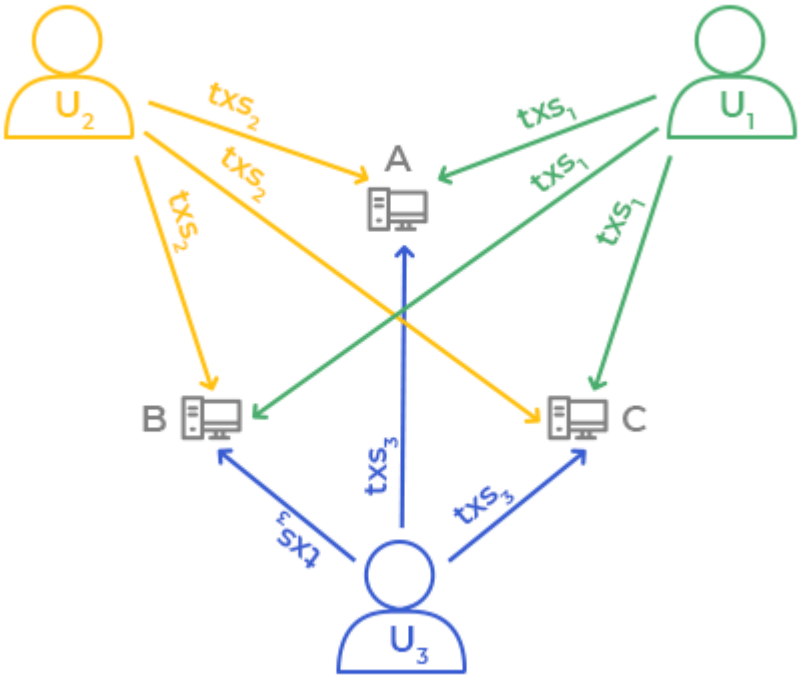
If the system splits into s shards and i IMIN transactions were recorded in block B_{n-1}, then each shard committee consists of $\frac{i}{s} = m$ minters. The size of the window of stakeholders selected in block B_{n-2} is algorithmically adjusted to maintain the required number of shards with constant m. In Dynemix, $m = 10$ (although it may be subject to adjustments if needed).

Determining m is a security/overhead trade-off. The more m we set, the more traffic overhead each minter node will suffer, which makes sharding less efficient, but at the same time, the more security will be provided, as the chance of controlling the supermajority in the shard for the adversary statistically decreases.

4) Collecting transactions

After the shard committee is appointed, minters collect incoming transactions.

Transactions are assigned to particular shards according to the public keys of the issuers (in the case of registration transactions, according to the username). Upon receiving block B_{n-1}, each node in the network knows to what shard committee any given transaction in the pool is assigned.



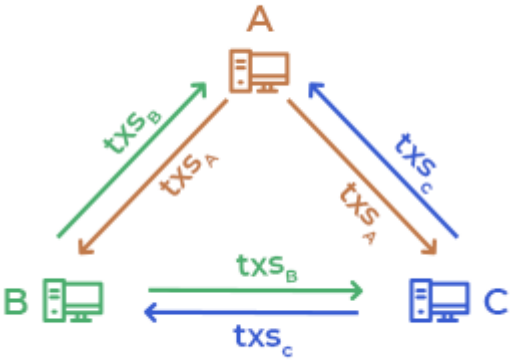
To ensure fast verification, transaction issuers or other nodes that store transactions should send them directly to all minters of the corresponding shard. If the nodes behave accordingly, under perfect conditions all minters of the shard committee should have the same set of transactions by the end of the subslot.

Still, even if the transaction senders try to multicast transactions to all assigned minters, due to the network delays and imperfect time synchronization, the sets may partially vary. To ensure the consistency of the transaction set, minters synchronize the collected data.

5) Synchronizing the transaction set

During this phase, each minter multicasts their collected data to other members of the shard committee.

Each minter signs the hash of the collected transaction set and submits the set to other members of the shard. By the end of the time slot, the transaction set should become consistent within the shard.



Minters may behave Byzantine and send different versions of the transaction sets to other members of the committee or submit no data at all. This situation is managed in the next phase.

6) Verifying each other's transaction commitments

To reach a Byzantine agreement, honest nodes should possess identical transaction sets. Since Byzantine actors in the previous phase can act inconsistently, an additional round of data exchange is required.

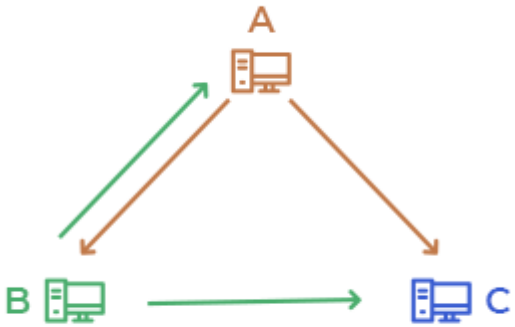
During this subslot, minters mutually exchange hashes of all the transaction sets received from other minters to verify that each member of the committee has the same total set available and that no one is trying to commit fraud.

If all minters are honest and non-faulty, each will get the same number of identical hashes, which means that the transaction set is consistent within the shard and that minters can proceed to the next phase. If there are Byzantine actors among the minters, however, additional actions are required.

Assume a simplified model wherein we have a shard committee of three minters. Alice and Bob are honest, Chuck is Byzantine, and consensus is reached by $\frac{2}{3}m$.

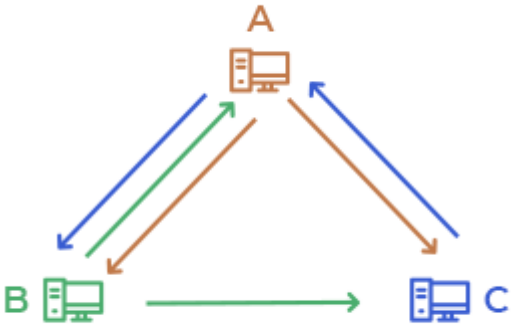
a) Chuck commits nothing

Alice and Bob exchange messages stating that Chuck has not committed anything. Since they exchanged their transaction sets in the previous phase, they can proceed to the next phase.



b) Chuck commits a set to Alice and nothing to Bob

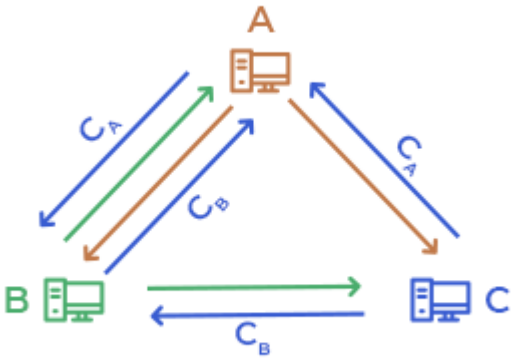
Bob receives a hash of Chuck's set hC from Alice, and since he did not receive the set from Chuck directly during the previous phase, he requests Chuck's set from Alice. After Bob receives it, the transaction sets become consistent, at least between Alice and Bob.



Bob may suspect that Chuck is an adversary, but since Chuck's malicious intentions are not provable, as he may simply have experienced connection problems in the previous phase, Bob does not apply any punitive measures.

Alice may suspect that Chuck is an adversary, but from the other side, Bob could also be an adversary trying to frame Chuck. Alice cannot reliably determine whether each of the assumptions is true, and she does not apply any punitive measures either.

c) Chuck commits sets to Alice and Bob, but the sets are not identical



Bob receives a hash of Chuck's set hC_A from Alice and finds that it differs from the hash of Chuck's set hC_B that he has received from Chuck. Bob requests Chuck's set C_A from Alice and Alice, having come to the same conclusions, requests C_B from Bob. After they exchange Chuck's sets, the transactions sets become consistent between them.

Now, as Alice and Bob both know that Chuck is the adversary, each of them creates a special proof-of-fraud transaction, accusing Chuck of fraud, with both signed hashes of Chuck's sets (hC_A and hC_B) attached as a proof, and adds it to the transaction set. If the shard-block proposed by Alice and Bob wins the consensus round, Chuck's stake is slashed.

The simplified model described does not take into account the possibility of multiple minters' being controlled by the adversary, who can abuse the situation described in paragraph **b** by sending messages to some minters too slowly, thus keeping the transaction set inconsistent by the end of the subslot.

To counter this opportunity and prevent the adversary from delaying the consensus, another rule is adopted – minters request the set only if it is known by at least $f + 1$ other participants by the beginning of the subslot. Otherwise, the set is ignored by all committee members.

7) Shard-blocks and the shard consensus round

After all minters within the shard synchronize the known transactions, each of them processes transactions and builds a shard-block that contains the whole synchronized set.

Minters exchange signed hashes of the shard-blocks to reach a consensus. Dynemix uses a synchronous authenticated multivalued Byzantine agreement model. If the required threshold of replicas proposes identical shard-blocks within the time subslot, this block is considered approved by the shard committee.

Minters can propose only one shard-block during this phase, which is why they should build the same block as most other minters probably will; otherwise, the consensus will not be reached, and nobody will receive a reward. The optimal strategy to reach consensus is including all known transactions in the shard-block.

If a minter proposes two different shard-blocks, he or she is accused of fraud by the honest minters and his or her stake is slashed.

In Dynemix, the shard consensus threshold is set to $> \frac{2}{3}m$.

As a Byzantine agreement protocol used in Dynemix can tolerate f faulty players of $n > 2f$ participants in a (weakly) synchronous communication setting, the required majority value c in a committee of m minters can be set within $\frac{m}{2} < c \leq m$. Setting c is a safety/liveness tradeoff. The more c we set, the more resources the adversary must possess to approve a deviating shard-block or fork the blockchain if synchrony is not held, but, at the same time, the fewer faulty minters can be tolerated. We have chosen $c > \frac{2}{3}m$ for shard consensus as an estimated point of balance. This value can be changed within the stated boundaries if needed.

8) Shard-block exchange

After consensus is reached in the shard, minters establish connections with other shard committees and mutually exchange patches of shard-blocks (before the quilt layer is implemented, minters will exchange complete shard-blocks) with them. After the exchange is complete, all external minters request hashes of the shard-block from all shard committee members to ensure that nobody behaved in a Byzantine way.

If Byzantine behavior was detected (e.g. somebody signed different versions of a shard-block), honest minters issue a proof-of-fraud transaction in the next block to penalize the malicious actors.

9) Master-block and master consensus round

After each minter has collected all patches (or complete shard-blocks), each builds a quilt and constructs a master-block. Minters exchange hashes of the master-block to reach the second BA agreement.

The master consensus threshold in a minter committee of M members is set to $\frac{3}{5}M$.

V. Dynemix Explained: Features And Tradeoffs

Now that we have briefly described the architecture of Dynemix, we should explain why we implemented the mentioned solutions and how they help us reach the stated goal of building a next-generation blockchain payment system, as well as what tradeoffs we must accept on the way to this goal.

1. Setting

Dynemix was developed to operate in a specific setting, which should be outlined to clarify certain features of the protocol.

In the beginning of the white paper, we stated a set of properties that we strive to obtain. One of the essential features is a high level of decentralization, which requires the following condition:

- **The system should be designed to let the maximum number of users be involved in the system's support processes**

We intend to go back to basics and design a system that can be operated by common users with the help of home-class hardware, which corresponds to the original ideas of Satoshi that were embedded, but not embodied, in Bitcoin.

1) Honest nodes

The first difficulty we face is the need to revise the notion of an honest node.

In a conventional setting, honest nodes not only refrain from doing anything that goes against the rules but also refrain from not doing something that is expected of them. In relation to many BFT-style blockchain protocols, this refers to being constantly available at least for the nodes known as current stakeholders (or any other types of nodes that are expected to participate in a state transition). In the case that an average number of stakeholders who are offline exceeds the threshold, such protocols will eventually lose their liveness.

The developers of Algorand relaxed these requirements and introduced the notion of "lazy honesty," which refers to users who honestly follow all their prescribed instructions when they participate in the protocol but are asked to participate to the protocol only very rarely (e.g. once a month) and with advance notice.

This notion does not fit our setting either. If we assume that users can participate in minting with home PCs, we cannot expect them to hold any particular uptime – users can arbitrarily go online and offline at any moment, regardless of being selected for minting in any particular round, which is why we need to relax the boundaries of availability even further and operate with the notion of *irresponsible honesty*.

Honest but irresponsible nodes are actors who do not commit any actions that deviate from the protocol but who can stop participating at any arbitrary moment and hold any arbitrarily long (up to infinite) period of unavailability.

> The described notion seems to have a lot in common with *the sleepy model of consensus* researched by Pass and Shi in their eponymous paper. Their model is abstract, however, and does not allow to easily derive particular practical conditions for which we are designing the protocol. To avoid confusion, we use a different term, which also places more emphasis on the partially subjective nature of the problem.

Although the notion of irresponsible honesty seems to fit our setting, to provide a more precise description of the targeted practical environment, we should further refine the model and introduce the notion of *weakly responsible honest nodes*.

In terms of Dynemix, weakly responsible nodes are actors who can arbitrarily refrain from participating even if asked to participate, but who, once accepted, hold some expected uptime, during which they perform all of their instructions. We assume that the more time that passes since the node committed its consent to participate, the more likely it is to lose availability.

What does this all mean in practice? We assume that the vast majority of users will use the Liberdyne messenger (or any other third-party client that operates in approximately the same manner) as a client for interacting with the Dynemix network. Once a user obtains enough dynes to place a minimal stake and his or her hardware satisfies the requirements, the client engages in minting.

After the stake is placed, we do not expect an average user to hold constant availability. It is obvious that, given that the client is run at home, the user can simply close the app or shut down the PC at any time. Under such assumptions, it is easy to conclude that practically any arbitrary uptime/downtime proportion is possible, which makes it look like the model of *irresponsible honesty* (as described above) or *the sleepy model* (as defined by Pass and Shi).

We also rely on another assumption, however, which brings us to the model of *weak responsibility*. Despite the user's ability to go offline at any time normally, in case he or she is selected to participate while being online, we expect the user to prevent losing availability and refrain from quitting by choice. In practice, this means that once the client figures out who has the right to participate in the creation of the next block, it alerts the user and prevents any kinds of soft shutdown until all the duties are carried out. The user is incentivized to wait, which makes us rely on his or her not closing the client until the time slot passes. At the same time, the more time passes, the less the user may have the opportunity to wait, if the need to shut down is urgent. Of course, this does not refer to situations when connection is lost due to a network or hardware malfunction, but this is expected to occur very rarely.

According to the described model, we expect the following distribution of participants:

- **Weakly responsible nodes** – the majority of participants. This category is represented by low-stakes users, who will likely engage in minting via their home/office hardware. The presence of such actors will greatly benefit decentralization.

- **Responsible nodes** – a minority of participants. Stakeholders who have placed stakes large enough to secure frequent participation will likely hold constant uptime. At a certain point, it will become economically feasible to run a node in a data center to reduce omissions.

- **Irresponsible nodes** – the smallest minority of participants: those who lose connection during participation. We assume that a small share of such nodes will be also present, as faults are inevitably expected to occur.

This model is inherently easily managed by Nakamoto-style protocols; it is not so trivial, however, to cope with it for a BFT-style linearly consistent protocol. To conform to the described model, the following measures were applied:

a) IMIN transactions

After a stakeholder is chosen to participate in block creation, he or she is expected to confirm his or her consent by issuing a special IMIN transaction directly prior to the process start. This measure circumvents each participant's initial lack of awareness of each other online status. After observing a set of IMINs in block B_{n-1}, all participants come to a consistent view on the set of the selected stakeholders who are actually available at the time, which prevents offline nodes from affecting liveness.

We assume that there is always a known average proportion of available and ready stakeholders with only some insignificant deviation. If the statistically determined proportion does not hold within tolerable boundaries of variation, it may negatively affect system throughput; i.e. if the number of the collected IMIN transactions is much lower than expected, the system will split into a smaller number of shards. In most cases this will have no consequences; if the system is running at the capacity limit, however, some transactions may not fit in and will be left in the pool.

IMINs help maintain liveness in the presence of weakly responsible nodes; the question of safety, however, is ambiguous.

On one hand, we assume that the adversary keeps his or her nodes always available and ready to fulfil his or her sinister plots, which is why non-responsible nodes who refuse to participate despite being chosen contribute to a higher relative number of adversarial nodes in the minter committee.

On the other hand, providing an opportunity to behave in a weakly responsible manner without any penalties contributes to a larger global stake pool size, which in turn decreases the relative share of the adversary and reduces his or her chance of being selected for minting.

It is hard to predict which of these features will outbalance the other one in practice.

Although IMINs provide a solution to the problem of weak responsibility and assure stronger liveness guarantees, at the same time they can negatively affect liveness from the other side. In case no IMIN transactions are added to block B_{n-1}, a minters' committee for block B_n cannot be formed, which terminates the protocol's execution.

Minters are incentivized to process IMIN transactions (minters of block B_n distribute rewards to minters of block B_{n-1}, which is why minters of the current block are always interested in the creation of the next block), hence, considering the redundant number of minters for each block, the mentioned situation is extremely unlikely to happen. If this issue is considered a feasible threat, it is possible to implement an algorithm to resolve it.

It is important to emphasize that IMINs as a solution for weak responsibility can only be implemented into a protocol that can guarantee a fair block proposal, which is, in relation to the issue at hand, adding all known IMIN transactions into the next block regardless of the proposer's preferences. If the protocol uses a conventional block-proposal algorithm, a malicious actor can simply censor the IMIN transactions of other stakeholders, thus permanently seizing control over the system. In Dynemix, this issue is resolved by our novel consensus solution called Guess My Block game, which is described below.

b) **Constant reshuffling of minters**

We assume that after a minter issues an IMIN transaction, he or she is expected to reliably hold availability only for a short period. For this reason, he or she is obliged only to participate in the creation of a single block and the subsequent reward distribution phase.

c) **No leaders or other single points of failure**

The Dynemix protocol is completely leaderless. If we assume the presence of irresponsible nodes, it is obvious that leaders can also behave irresponsibly. If we apply a view-change procedure in case a single shard leader fails, it will either lower the performance of the entire system or simply prevent consensus in the shard, which opens up an opportunity for an attack on liveness or fairness.

d) **More important actions first**

We assume that the more time that has passed since a minter issued an IMIN transaction, the less we can rely on his or her availability. As the protocol execution advances within each round, this fact is taken into consideration.

2) Time

Dynemix operates in a setting with *synchronized clocks* adjusted to *the global time*. The time is divided into discrete slots of 10 seconds (this is subject to adjustment if required), which each correspond to a block created during that time slot. A time slot is split into a number of subslots corresponding to the phase of the protocol.

We assume that all nodes, once connected to the system, synchronize clocks and have an approximately consistent view on the current protocol execution phase. Nodes are allowed to experience minor discrepancy in their local views on the current time, given that this discrepancy is insignificant on the scale of the determined subslots.

Known time synchronization algorithms for peer networks can be classified into two categories:

a) **Symmetric (peer-to-peer)**

In symmetric protocols, each participant has equal influence on the system time. This approach can be used for blockchains mostly in a limited-view manner, which corresponds to communication overlay of various peer networks, including blockchains.

Intuitively, the symmetric approach seems to be the most relevant to a blockchain system, as it satisfies the principles of decentralization; it also, however, has several shortcomings.

The first one is its highly questionable Sybil resilience. The attacker can spread his or her infected view over the system, thus massively breaking consistency, or can perform an eclipse attack targeting particular nodes. Although there are several proposed techniques for the Byzantine setting, they may not be robust enough when the liveness and consistency of a blockchain is at stake.

Another problem is scalability. We assume that peer nodes cannot have a complete view of the system and are limited to a local view, which consists of other nodes with which the node can interact. As the system scales up, the relative size of the local view of each node shrinks, which makes the protocol more vulnerable, especially in the presence of a powerful adversary.

b) **Asymmetric (client-server)**

The mentioned shortcomings of symmetric design can be mitigated by moving to an asymmetric scheme. In asymmetric design, specially appointed nodes serve as master-servers for the rest of the network (or a more complex multi-level hierarchy can be used), which allows the number of time-setting servers to be limited as desired.

> Asymmetry solves the problem of scalability, but providing more resilience still requires additional measures. The obvious solution would be binding a resource to the time-setting powers of a time server, which can significantly hamper a Sybil attack without centralizing the process. In relation to Dynemix, this means creating Proof-of-Stake time servers. This will likely require a sophisticated solution, however, which would require additional research and obviously would not contribute to better overall performance.

Having assessed all pros and cons, we conclude that currently the most optimal solution is simply to use public NTP infrastructure for time synchronization. This solution can reasonably be criticized for relying on an external source, which does not seem appropriate for a truly decentralized system; it provides certain benefits, however, that make this decision much less controversial:

- NTP is the most popular time synchronization solution to date. It has existed for decades and tested against various attack vectors. Given that scalable time-sync protocols for blockchains (and peer networks with Byzantine actors in general) are still in their infancy, it may be reasonable to turn to a more reliable solution, at least until a robust decentralized protocol is designed and tested.
- Engaging NTP will bind the system to the external infrastructure to a certain extent, which can provide more robustness. Given that time sync will be left out of bounds of the blockchain protocol, it will not be possible to distinguish the particular fraction of NTP infrastructure that is used by the system. For this reason, the adversary will not be able to aim specifically at Dynemix, but instead will have to attack the entire NTP infrastructure, which is used by a huge array of various systems.

3) Communication

We use the Δ-bounded model of communication. Whenever each node sends a message, this message is received by a recipient node within Δ delay. All messages are cryptographically signed, which ensures authenticated communication based on the global PKI setup.

The protocol does not require constant strong synchrony and makes progress as long a threshold of designated replicas hold a bounded delay Δ during each round of execution. At the same time, nodes that temporarily lose connectivity are allowed to rejoin the protocol and restore consistency.

Essentially, we adopt the model of *weak synchrony* as described by Guo, Pass and Shi. We consider this model the most realistic description of the practical setting that we assume.

4) Lock-step and responsiveness

Dynemix is designed to operate in a *lock-step* execution manner, which means that the protocol makes progress in strictly predetermined time intervals, which are set according to the estimated communication latency, bandwidth and computation delay assumptions of the peer nodes. It may seem much better to achieve *responsiveness*, however. The protocol is considered responsive if the progress is made according to the actual communication and computation delays of the participants, independent of the expected upper latency bounds.

Responsiveness is thought to be a feature of asynchronous protocols. One can, however, envision a sophisticated hybrid solution that could bring responsiveness into the synchronous model as well. For example, in Thunderella (and several other recent solutions), the mentioned property is achieved by packing an asynchronous algorithm into an underlying synchronous algorithm, thus achieving responsiveness on the fast optimistic path and keeping the minority corruption tolerance on the slow fall-back path, which is engaged once the optimistic conditions are not met.

As we stated in the beginning, fast finality is one of the essential features of a decentralized payment system, which raises a question: can we upgrade Dynemix to achieve responsiveness? Since we are designing our protocol from scratch, we are not bound by any particular models or restrictions, and it looks tempting to apply some tricky solution that can help speed up the system.

As an answer to that question, we would say rather more "no" than "yes." There are several fundamental factors that deprive us of such an opportunity.

a) **All replicas need to wait for the end of the time slot to make sure that everyone had a chance to commit.**

   If we allow the protocol to proceed after obtaining a threshold of signatures at the shard consensus stage, slower minters will be deprived of the opportunity to participate and receive a reward. Such a situation will give an unfair advantage to those who run nodes in fast data centers on powerful hardware and will disrupt the system of essential economic incentives of the protocol.

   If we tighten the optimistic conditions from obtaining a threshold to obtaining all signatures, given the total number of participants, it is highly unlikely that not a single one of them would experience a communication fault or simply behave unresponsively, which means that such optimistic conditions will happen so rarely that it barely makes sense.

   A similar situation occurs at the master consensus stage. We need to assemble the most complete set of shard-blocks, which means that we can either proceed only after all expected shard-blocks are propagated or wait until the respective subslot expires.

   Moreover, even after obtaining all signatures or collecting all shard-blocks, it is still preferable to wait until the end of the time slot to assure the opportunity to present a proof of fraud in case adversarial behavior is detected. From that perspective, responsiveness makes the system less secure.

b) **Time slots should be consistent among all nodes, including those who currently do not participate in the consensus.**

   Most BFT protocols separate the participants of the state transition procedure from the external nodes. For example, in PBFT, an external node (denoted as a client) sends an input value to one of the participants (denoted as the leader), who then reaches a consensus with the other participants (denoted as backup replicas) and provides the agreed output value to the client.

   In variations adapted for blockchains, the leader assembles a block from transactions that he or she somehow obtained beforehand, drives a consensus with other replicas and propagates the resulting output-block through the network. Commonly, input values are propagated through the network via a gossip protocol, which ensures that each time a node is selected as a leader, it possesses a large chunk of currently pending transactions. A node that issues a transaction does not need to send it to particular nodes at a particular time – gossip assures that the transaction will get to a leader with an expected modest delay with high probability.

   Dynemix, on the other hand, features a completely different input value-submission algorithm. A critical condition for it to work properly is that the client submits transactions to a predefined set of replicas during the prescribed time slot. All nodes in the system (including both clients and replicas) should have a consistent view on these subslots for each round of protocol execution, which contravenes the notion of responsiveness.

   This algorithm is implemented to achieve the crucial property of conventional centralized payment systems: impartial instant processing of all transactions that were sent to the system, which is undoubtedly a higher priority than achieving responsiveness.

2. Censorship and transaction fees

In the beginning of the white paper, we mentioned two important features of a decentralized payment system that we intend to achieve:

- **The system should instantly process all transactions that users send to the network.**

- **Transactions should be processed without fees.**

These features will help raise the level of user experience to the level of existing conventional payment systems, which is an essential condition that can allow a blockchain platform to compete with those systems on equal footing.

Most current generation blockchains (whether of a PoS or PoW design) rely on a leader who proposes block candidates. Other nodes vote for the proposed block and the network either accepts or rejects it.

According to this approach, the leader assembles the block at his or her own discretion and is free to reject any transactions. He or she may consider the proposed fee insufficient or may simply have personal feelings about the transaction issuer. For this reason, users must compete for the leader's favor (in most cases by offering a higher processing fee), which severely deteriorates the user experience compared to conventional payment systems, as they process all transactions without any preference. Furthermore, this approach creates the preconditions for censorship.

Such an approach is especially dangerous for Dynemix, since we have implemented a solution for maintaining liveness in the presence of weakly responsible nodes (the described above IMINs) that strongly relies on impartial block proposal. The ability to censor transactions can lead to the capture of the system by the adversary.

Although there is a number of projects that addressed the problem of censorship (for example, Honey Badger introduced multi valued transaction set proposal), to date no one managed to get close to the stated parameters.

As we intend to process transactions for free, it complicates the task even further – why would the leader put any particular transaction into the proposed block, if he or she has no incentives for doing so? This issue may seem insoluble within the currently available concepts, and to date, we have seen no solution that can help achieve both stated features assuming the non-altruistic behavior of participants. That is why we had to develop a new design of a block proposal algorithm and introduce a completely different approach.

3. Guess My Block game

1) Brief description

Our solution to the problem is offering minters to play a game with incomplete information, in which the optimal winning strategy implies adding all known transactions to the proposed block and the resulting consensus between players is reached via a synchronous authenticated multivalued Byzantine agreement.

In Dynemix, no leaders are appointed in shard committees. All minters propose their own shard-blocks on equal footing and have equal votes during the multivalued consensus round. The voting is carried out not by accepting or rejecting a shard-block proposed by the leader, but by proposing identical or non-identical shard-blocks by players.

✅ ***If the majority (supermajority) required to reach a Byzantine agreement propose identical shard-blocks, their blocks win and all minters who proposed those blocks receive a reward. The minters who proposed different versions of the shard-block or proposed nothing are not rewarded.***

❌ ***If identical shard-blocks are proposed by an insufficient number of minters, a consensus is not reached, the shard-block is skipped and no one receives a reward.***

If someone proposes multiple different versions of a shard-block (behaves Byzantine), his or her vote is not counted by honest minters and his or her stake is slashed.

This game implies the need to guess the block that most of the minters will propose without being aware of the preferences of all the others. The optimal winning strategy is to commit a block

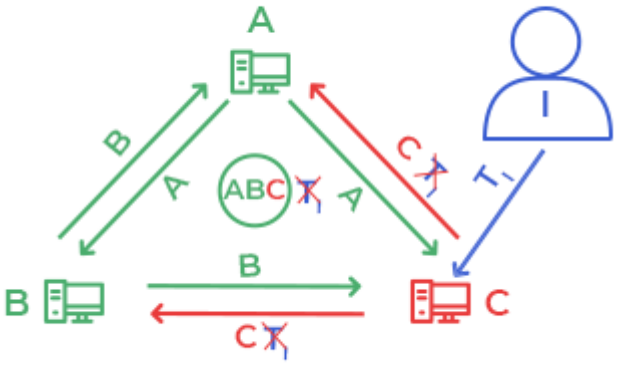
assembled according to the default rules, which means including all known transactions (or any other set of rules that is reliably expected to be supported by the majority of players).

2) Game explained

Assume that Ivan is a transaction issuer and that the shard committee consists of $m = 3$ minters, where Alice and Bob are impartial rational players and Chuck does not like Ivan and intends to deny his transaction. At the same time, none of the minters can coordinate his or her actions with others, Ivan is not aware of Chuck's intentions and a consensus in the shard is reached by $\frac{2}{3}m$.

Also assume that all messages between the participants are delivered within the expected Δ-boundary and that nobody behaves Byzantine (the Dynemix protocol can tolerate communication faults as well as Byzantine behavior, but to simplify the model these conditions are neglected in the following description).

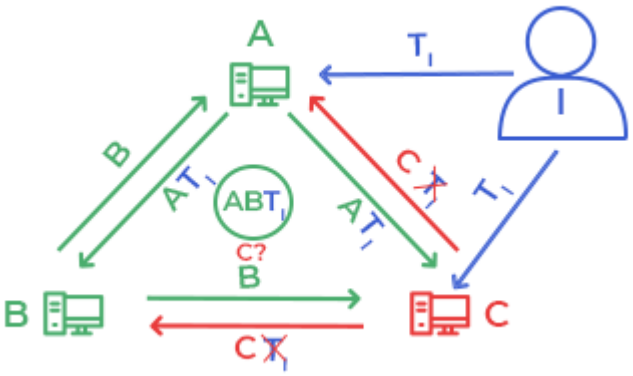
a) Ivan sends the transaction only to Chuck.



Chuck receives Ivan's transaction T_I but has no intention of adding it to the block. For this reason, during the transaction sets exchange phase, he submits his set to Alice and Bob, but does not include T_I. He receives Alice's and Bob's sets and sees that they are not aware of T_I. Since none of the minters included T_I into their sets, they all propose identical shard-blocks without T_I and each of them gets a reward. Chuck wins.

For this reason, Ivan should send his transaction to all minters of the shard.

b) Ivan sends the transaction to Alice and Chuck.

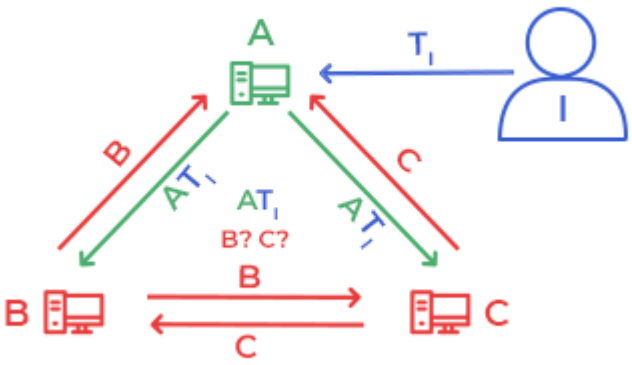


Chuck receives Ivan's transaction T_I but has no intention of adding it to the block. During the transaction sets exchange phase, he submits his set to Alice and Bob but does not include T_I.

He receives Alice's and Bob's sets and learns that Alice was aware of T_I, while Bob was not. Since Alice sent Chuck a set with T_I, Chuck assumes that she sent the same set to Bob.

During the hash exchange phase, he receives the hash of the Alice's set hA from Bob, and since it is identical to the hash of the set that he received directly from Alice, he is now sure that Alice and Bob are both aware of T_I. Chuck knows that if Alice and Bob propose identical shard-blocks, they will win the consensus round regardless of his decision. He has no choice but to commit a block with T_I included if he expects to receive a reward.

c) **Ivan sends the transaction only to Alice, but both Bob and Chuck dislike Ivan.**



Now the game gets complicated. Not only does Chuck want to reject Ivan's transaction, but Bob also intends to do the same. At the same time, Bob and Chuck are independent minters, their actions are not coordinated and they are not aware of each other's censorship preferences.

Even if Bob and Chuck received T_I, neither would include T_I in their sets, but to their misfortune, Ivan sent the transaction to Alice, who is impartial. She includes it in her set, and after the exchange phases, Bob knows that Alice and Chuck are assuredly aware of T_I, and Chuck knows that Alice and Bob are assuredly aware of T_I.

Both Bob and Chuck would prefer not to include T_I in the block, and if they propose identical shard-blocks without T_I, they will win the consensus round, T_I will be rejected, Bob and Chuck will receive a reward and honest Alice will be left without a reward.

Since they are uncoordinated, however, they cannot be sure of each other's intentions (meaning their mutual agreement not only to reject T_I but also whether to reject or accept any other transactions). Chuck may suspect that Bob also dislikes Ivan and might be willing to reject T_I, for Bob did not submit T_I to Chuck in the set exchange phase, but this is a too weak assumption to rely on, unless Ivan is the most hated person in the world.

Chuck and Bob may turn to the rushing strategy: that is, holding their shard-blocks until they see all the other proposed blocks and committing their blocks afterward. If they do so, however, both will only get the hash of Alice's shard-block with T_I included and eventually each of them will be forced to make one of the following decisions:

- To wait more and risk missing the moment when the time slot expires, and thus not getting a reward.
- To commit a shard-block with T_I, thus winning the consensus round together with Alice regardless of the decision of the third player and getting a reward.
- To commit a shard-block without T_I and hope that the third player will side with him and commit an identical block.

The third option is obviously non-optimal, since it relies on a probabilistic assumption that the same censorship preferences will be supported by the consensus supermajority. To a certain extent, this assumption may be eligible with a single T_I and only one remaining waiting player, whose decision will determine consensus, which means that the voting simply turns binary.

If there are ten minters in the shard, however, and Chuck intends to reject the transactions of different issuers (say T_I and T_D) simultaneously, he cannot rely on others having exactly the same preferences. Some may agree with rejecting T_I, but at the same time, they may not agree to reject T_D, which is why they may prefer to side with the honest minters.

Since the voting is not binary and implies a huge number of potential proposals, no reliable assumptions on the censorship preferences of other players can be made, and no unitary adversarial strategy under the uncoordinated adversary assumption can be predicted.

When each rational minter makes a choice, he or she presumes that there can possibly be up to $m-1$ honest minters (ones who follow the default rules, in our case) and up to $m-1$ adversaries, which makes the odds even, but at the same time, honest minters will propose only one variation of a shard-block, while the adversaries can possibly propose up to $m-1$ different variations chosen from 2^t combinations, where t is the amount of all known transactions, making including all transactions into the block the optimal strategy for both rational and altruistic players. The default known censorship rules may be considered a focal point of the game.

3) Game's guarantees

Guess My Block game ensures with high probability that each T_n will be added to the next block, given that both of the following assumptions are true:

- T_n was received by at least one player, who does not intend to reject T_n.
- Players who intend to reject T_n do not form a coordinated consensus threshold.

Guess My Block may not be efficient if the overwhelming majority of minters start arbitrarily rejecting transactions from issuers, ultimately reducing their sets to zero. This threat is less relevant to blockchains that feature commissions for transaction processing, but this may cause a problem with the free-transaction design of Dynemix.

The fundamental limitation on the way to creating a solution for a fully rational environment is the inability to non-interactively prove that certain data were transferred from one node to another. As the transaction sender cannot prove that he or she actually tried to commit a transaction to the minter, we cannot apply any punitive measures on minters who refuse to add certain transactions to their sets.

On the other hand, with the free transaction concept, there are no incentives to include any transactions in the set exchange phase, which is why at least one altruistic committee member is required. Even considering that we designed Dynemix in a way that allows for the participation of common users, who are less likely to break the protocol rules without a really strong reason, we still find it reasonable to add an additional incentive to include more transactions in a block.

4) Additional incentive

To ensure the appropriate behavior of rational participants, we added a coefficient that increases rewards for minters of those shards whose shard-blocks include more transactions.

Since the rewards for block B_n are distributed throughout block B_{n+1}, the minters of B_{n+1} count balance changes included in the patches of B_n and calculate the average number of transactions in the shards. According to the calculated value, a coefficient is applied to each minter and the reward is adjusted accordingly.

The correlation between the number of included transactions and the size of the reward is sublinear. Otherwise, it would create incentives for spam-transactions – minters could create Sybil

accounts and add their own meaningless transactions to their sets, only to increase the overall number of transactions in the shard. This issue is also resolved by limiting the size of a minter's transactions set, which disallow the adding of an arbitrary number of spam transactions.

Minters are indirectly incentivized to behave correctly, as the value of their rewards depends on proper system operation, which is why we consider a moderate reward adjustment sufficient.

5) Guess My Block in the master consensus round

Master-blocks are assembled according to the principles of Guess My Block as well. Instead of transaction sets, minters exchange patches and shard-block headers (or complete shard-blocks) to collect the most complete set.

Minters are rewarded only in the case that a shard-block signed by them is included in the master-block, which incentivizes all members of each shard committee to spread their block as widely as possible.

Considering that each rational minter will obviously vote for a version of the master-block that contains his or her own shard-block, the most complete set becomes the focal point.

During voting, the shard committees are disbanded, and each minter behaves independently. The consensus threshold in a minter committee of M members is set to $\frac{3}{5}M$.

> With the growth of the system, a master consensus round will start creating a larger communication overhead. Unlike shard-committees, which feature a fixed number of participants, master-committees scale according to the size of the account base and the average TPS. At some point, the number of replicas will reach the thousands, which raises the question of further optimization.
>
> In this case, it is reasonable to appoint a number of delegates form each shard, who will be authorized to participate in a master consensus. The largest stakeholders among the minters who signed the shard-block seem to be the optimal choice.

It can be clearly observed that at this stage we apply a lower threshold than during a shard consensus. By doing so, we respect the weakly responsible model described above. We assume that some minters may not make it to the master consensus, which is why the liveness threshold at this stage should be slightly relaxed. In addition, validity cannot be breached at this stage (in the meaning of state output), as all shard-blocks are already settled by the shard committees, which is why we consider it an appropriate measure to provide more liveness at the cost of consistency guarantees.

One may ask a reasonable question – why can we not simply set the threshold to $n > 2f$, which is a known lower bound for a weakly synchronous communication model? Indeed we can. As we are developing a protocol for practical implementation, however, we are not restrained by the assumptions of any particular model, and we can take different approaches if we consider them to meet our needs.

Unfortunately, protocols that tolerate minority corruptions can tolerate zero corrupted replicas under asynchrony, which is a fact that concerns us. Although we rely on weak synchrony, we still respect the probability of some extreme situations in which the system can be partitioned due to a massive network failure (caused by a war, a global cataclysm etc.). In this situation, we would prefer to retain at least certain minor consistency guarantees that can assure an acceptable level of safety.

With the chosen threshold, the system can hold consistency in the presence of up to $n > 5f$ corrupted nodes even under asynchrony, which seems sufficient given the very low probability of the occurrence of asynchrony, especially with the presence of a powerful adversary who can control communication.

With the help of the described algorithm, we achieve the following:

- assurance that all assembled shard-blocks will be included in the master-block;

- prevention of the grinding of different combinations of shard-blocks to find a random oracle that favors the adversary.

It is worth noting that during the master consensus round, there can be a short opportunity for the adversary to try a grinding attack. To be able to grind a hash of the master-block, the adversary should meet the following conditions:

- control the BA threshold of minters in at least one shard;
- receive all other shard-blocks before grinding.

If the adversary fulfills both requirements, he or she will have a very short time slot to grind different versions of a shard-block in order to find a master-block with a hash that will provide him a better representation in the next minter committee.

We do not consider this attack vector to impose a serious threat for the following reasons:

- Obtaining a consensus threshold in even a single shard is not easy to achieve, unless the adversary controls a comparable proportion of stake in the global pool. In other cases, he or she will gain an opportunity to take over a shard only extremely rarely at best.
- A time window for grinding is so short and the number of possibilities is so huge that the probability of finding a hash that can provide significant advantage is neglectable even for a very powerful adversary.

6) Coercive attack

Although the adversary is required to possess $> \frac{2}{3}$ (the consensus threshold) of the total voting power to arbitrarily apply any desired censorship rules, the adversary who controls $\geq \frac{1}{3}$ (the liveness threshold) of the voting power can try to force rational minters to follow his or her censorship rules under the threat of halting the consensus and preventing minters from receiving rewards.

An attack can be executed in different ways, but the basic conditions are the following:

- The adversary controls $\geq \frac{1}{3}$ of the voting power and convinces the majority (or other required threshold) of the other stakeholders of the seriousness of his or her intentions;
- The adversary publicly declares his or her block assembly rules and actually follows them.

If all participants except the adversary are purely rational, they will conclude that if the adversary always proposes shard-blocks assembled according to his or her declared rules, he or she will either win or break the consensus. The only option to get a reward for others is to follow the declared rules.

We do not consider this attack scenario realistic in the practical setting of Dynemix for two reasons.

First, the adversary will unlikely be able to convince the required threshold of honest stakeholders to submit to his authority and let him control the system in this way. It is more likely that honest minters will reach an agreement and simply ban the adversary via a hard fork. Considering the size of the stake required for the attack described, the adversary will suffer tremendous damage.

Second, such behavior is not completely rational from the point of view of the adversary himself. Considering the setting of Dynemix, the adversary cannot expect all minters to behave purely rationally. Instead of forcing others to follow his rules, he may end up simply stalling the protocol, which is against his own interests.

If we assume that the adversary is irrational and actually intends to harm the system, we can do nothing against a malicious actor who controls $\geq \frac{1}{3}$ of the voting power, as it is an inherent liveness threshold of any BFT consensus model based on an honest supermajority assumption. The only solution is to go beyond the protocol rules and slash the stake of the adversary via a hard fork.

Guess My Block is designed to operate in Δ-bounded (weakly) synchronous setting, which means that it can actually tolerate up to f corruptions out of $n > 2f$ replicas as long as synchrony holds for the required threshold of replicas. We can take advantage of this property and simply set the shard consensus threshold to $> \frac{1}{2}m$, which renders the described attack pointless, as it would require almost the same amount of resources as a consensus takeover attack.

This will reduce safety guarantees, however, which we consider a higher priority (especially in relation to validity). For this reason, we decided to keep the threshold of shard consensus at the level of asynchronous BFT protocols, assuming that the described attack does not pose a serious practical threat.

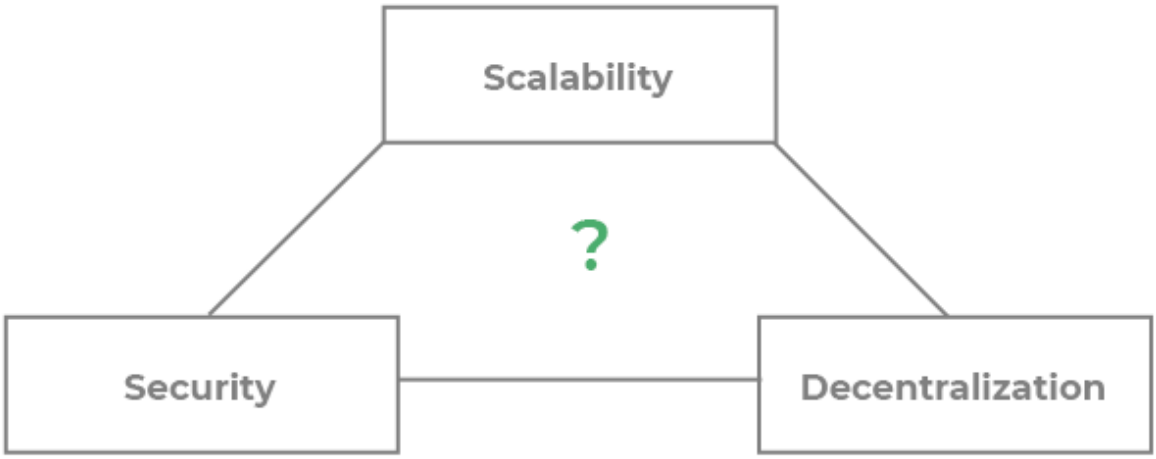
4. Scalability issue

In the beginning of this white paper, we stated an important feature of a decentralized payment system that we intend to achieve:

- **The system should be capable of processing at least 10,000 TPS.**

Most first-generation blockchains suffer from the scalability issue. Many developers of recent projects have addressed the problem and proposed a number of solutions, which include different tradeoffs.

The main problem at hand is a trilemma that states that higher scalability, security and decentralization cannot be reached simultaneously and that only two properties can be targeted.



Known solutions to the scalability issue can be conditionally classified into two categories, depending on which vertex of the triangle is sacrificed:

a) **Reducing the number of validators to a small group (sacrificing decentralization)**

This approach solves the problem by limiting a possible validator set to a small professional group that possesses powerful hardware and broad bandwidth, which allow them to process and synchronize large data flows. This helps significantly raise the network delay, bandwidth and computing power assumptions.

The set of validators can be limited directly by the protocol rules (DPoS) or by applying a high participation threshold.

Though such a system may be formally considered permissionless, in practical terms this approach makes the system semi-permissioned, as the participation opportunity is drastically

limited, which inevitably leads to the emergence of a coordinated oligopoly, whose interests include preventing new players from joining.

Unfortunately, following this approach leads to the severe centralization of the system, thus making it essentially pointless. This is a tradeoff that we cannot accept, which is why we had to search for another solution.

b) **Sharding the system (sacrificing security)**

Another approach is based on splitting the network into independent or semi-independent partitions, thus distributing the load between them respectively.

Though sharding may be considered an effective solution to the scalability issue, it negatively affects safety. Sharding makes the system vulnerable to a dangerous attack vector of a fraudulent block approval.

In classic blockchains, blocks, which are propagated across the entire network, contain all the data necessary to verify that the state change occurred according to the rules of the protocol.

If an adversary tries to commit an unsubstantiated state change (for example, adding to his or her balance an arbitrary number of tokens that emerge out of thin air), the adversary's deviating block will be rejected by all nodes. Since the adversary is disincentivized to make such attempts (in PoW systems, he suffers from the waste of resources, and in PoS systems with slashing rules he loses the stake), non-sharded blockchains are highly sustainable against this attack vector and feature strong validity guarantees.

The sharding concept stipulates partial data propagation to ensure the increase in scalability, which is why the security of the entire system is reduced to the security of each shard. Considering that cross-shard interaction is based on trust between validator committees, succeeding in one shard allows the adversary to affect the entire network.

As the stated approaches engender radical tradeoffs, we had to develop a non-trivial solution to maintain the desired level of decentralization without sacrificing security.

5. A brief introduction to sharding

Our sharding solution corresponds to the specific settings and limitations caused by the targeted properties we intend to achieve. To help better understand our approach, we will briefly describe known sharding concepts and issues from which each of them suffers.

Typically, we can use two different sharding concepts:

a) **Without the global state**

This approach isolates the shard state and hence requires additional cross-shard communication about the majority of transactions (particularly transactions that change the states of accounts assigned to different shards).

Though sharding with isolated states provides more scalability, it also begets more problems.

The first and most important issue is security. As the state transition is performed and validated only within the shard, a successful attack on a shard breaches the security of the entire system. The most dangerous vector in this case is adaptive corruption or coordination. Since validator committees in isolated shards cannot be quickly reshuffled, the only option to mitigate the risks is to appoint very large committees and assume that the attack delay will be assuredly greater than the reshuffle rate.

In addition to the security issues, complete sharding poses another serious problem – delayed finality.

The global state contains information only about shard assignments, and account states are stored only within shards to which accounts are assigned. Each shard works independently, and in the case of a cross-shard transaction, the problem of atomic commits arises.

Proposed solutions to the problem feature sequential updates of the states of both the sender's and the recipient's shards in four steps, which significantly increases finality latency.

Finally, complete sharding reduces data availability, which causes concerns, especially in relation to crucial data, such as the states of the accounts.

b) **With the global state**

Partial sharding involves the sharding of a state-transition procedure, but at the same time does not isolate shard states, assuming that all shard states are assembled into the global state, which is then propagated through the entire network.

This approach helps solve certain security issues, such as adaptive corruption and coordination, by allowing the reshuffling validator sets as quickly as needed. It still features lower overall security than non-sharded designs, however, due to the partial data propagation.

In addition, partial sharding allows us to solve the finality latency issue, and to increase availability of crucial data.

On the other hand, it provides weaker scalability.

6. Problems of sharding in Dynemix

1) Our sharding scheme

Having assessed all the pros and cons of the described designs, we concluded that currently the optimal solution for the scalability issue in Dynemix is partial sharding. We also find it feasible, however, to implement complete sharding architecture in the future. A possible solution would require further research, and it is more reasonable to start with partial sharding to assess the system behavior in practice under real-life conditions before moving toward the implementation of complete sharding.

The main reason for such a decision is that we do not treat scalability as a property that should be enhanced as much as possible. We need to reach only a presumably sufficient level, exceeding which does not bring any more tangible benefits. There is absolutely no sense in providing a million TPS if we assume that the demand can simply never grow that high.

According to our estimates, which are based on the available statistics on conventional centralized payment infrastructure, 10,000 TPS is the approximate upper level of demand under current conditions. Given that we are creating a potentially more advanced infrastructure than has been available to date, we may assume that the variety of use cases may expand in the future, pushing the demand beyond the stated boundary, but in any case it will unlikely be exceeded by an order of magnitude (especially considering that Dynemix is not designed to support Turing-complete smart contracts).

Partial sharding allows us to reach the required scalability while at the same time providing solutions to several important issues, which is why we consider this option preferable. Complete sharding should be considered only if we face a substantial lack of scalability of the current design.

In Dynemix, each transaction is assigned to a particular shard according to the transaction's issuer. Minters of the shard process the transaction and update the states of both the sender and the recipient (or multiple recipients if the transaction has multiple outputs). The resulting state changes are then propagated as patches throughout the network so that the minters of the next block know the entire state of the system and there is no need for cross-shard communication during the transactions' processing.

This approach allows transactions to be finalized within one consensus round and excludes the opportunity to abuse the train-and-hotel problem for double spending. With the help of a global state, we apply constant resharding, which helps instantly adapt to the current level of demand, provide a solution to the weakly responsible model of participation and resist adaptive corruption.

It still brings a number of security issues, however.

2) Adaptive attacks

In the section describing the Guess My Block game, we noted that the game puts a number of restrictions on the system design that negatively affect different properties of the platform.

One of those restrictions is the inability to involve a large number of participants. With the growth of the shard committee, communication overhead may become intolerable for most players, and scalability and decentralization will be negatively affected. This means that to provide sufficient scalability and keep the sharding solution efficient, we have to keep the size of the committee small, but this inevitably reduces security in the adaptive corruption model.

Assume that the adversary intends to bribe minters to approve his or her fraudulent transaction or censor certain third-party transactions. As we apply sharding, this means that, instead of bribing the majority of stakeholders from the global pool, the adversary needs only to choose one shard committee after its designation and bribe the consensus supermajority within a single shard.

As far as full nodes operate in the blockchain mode, the adversary can only commit a censorship attack. Validity is not jeopardized, for shard-blocks are additionally verified by all nodes in the system (in the same way as in non-sharded blockchains). Switching, however, to quilt opens an opportunity to approve an invalid state change, since validity stops being verified beyond a shard consensus. This threat is countered by the authorized master-nodes concept, which will be described further. Nevertheless, we still find it crucial to resist the corruption of shard committees.

Theoretically, this problem may be solved by appointing a very large committee (which will increase the attack delay assumption), but we cannot afford this due to the Guess My Block restrictions. We need to solve the problem in a different way.

Assume that we have a committee of ten minters, and given the BA threshold of $> \frac{2}{3}m$, the adversary needs to bribe only seven of them. This task looks feasible if the adversary has enough time to coordinate the minters' actions.

With the help of the global state, however, we can apply the constant reshuffling of shard committees to deprive the adversary of the opportunity to coordinate minters. As shard committees are formed only for the creation of a single block, the adversary only has time between the propagation of block $B_{n-1}$and the shard-block commitment of block B_n, which takes about 6–7 seconds. Bribing minters in such a short period of time does not seem possible, making Dynemix secure in the presence of a mildly adaptive adversary.

The same can be applied to adaptive coordination. If minters assigned to one shard committee try to coordinate their actions to approve a fraudulent shard-block that grants benefits to all attack participants, they will not have enough time to reach an agreement within the time slot at their disposal.

We can conclude that Dynemix is not designed to be secure in the instant adaptive corruption model. We do not, however, consider the assumptions on which this model is based practically attainable in the Dynemix setting.

3) Consensus takeover attack

One of the typical attacks on blockchains features an adversary who obtains the amount of resources required to affect a consensus. By controlling a liveness or consensus threshold of the participating replicas, the adversary can influence several properties of the system. In relation to

most decentralized SMR systems, we can talk about *validity*, *consistency* and *liveness* as the set of properties that can be compromised by the adversary.

The design of Dynemix allows to include an additional property in this list – *fairness* that is in our case, agreeing on an output derived from an uncensored set of inputs. In most blockchains, the set of transactions to be included in a block is chosen by the block proposer (or the leader), who acts at his or her own discretion, which is why fairness cannot be guaranteed during a state transition. With the help of Guess My Block, however, Dynemix provides certain fairness guarantees, which is why we can include fairness in a set of properties that can be held under normal conditions.

Another important feature of the platform is a two-layer consensus, which means that the stated properties can be actually violated twice during a single round of protocol execution. This makes the attack scenarios more variative and complicated.

Finally, Dynemix can function in the blockchain mode (meaning that all nodes operate with complete shard-blocks and master-blocks) or in the quilt mode (when only patches are exchanged, and a quilt is built during the master consensus phase). These two types of operation also feature different optimal attack strategies and adversarial thresholds.

In this section, we will describe the most dangerous attack scenarios and try to evaluate the level of resilience that can be provided with the current design.

First of all, we need to apply measures that impede the Sybil strategy.

a) Weighted sampling

The obvious measure is applying weighted sampling on a minters' committee designation stage. This helps provide a better chance of being selected for stakeholders with larger stakes, and hence creates an incentive to increase the stake size, which negatively affects the adversary who splits his or her stake between Sybil accounts.

Since shard committees feature a low number of participants, however, there is statistical dispersion that provides a non-zero chance that after a certain amount of rounds the adversary eventually may get lucky enough to have more than an average number of his or her Sybil accounts simultaneously assigned to a particular shard, which can lead to a single-shard consensus takeover.

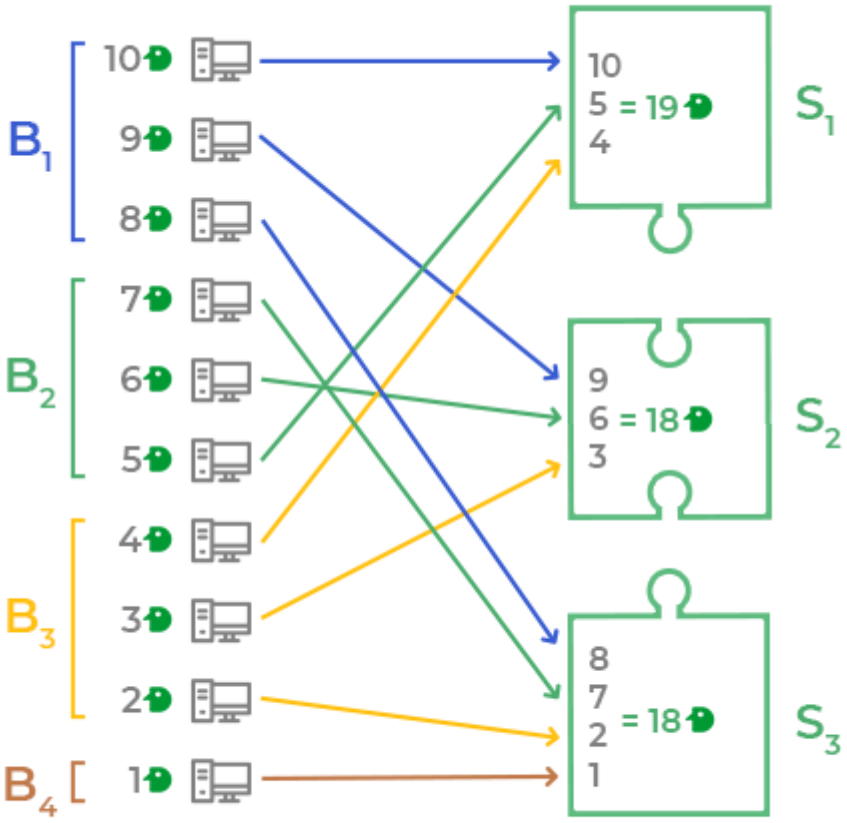
Under such conditions, if the adversary tries to control shards, the optimal strategy is to split the stake among the highest possible number of Sybil accounts (the total stake of the adversary divided by the minimal stake amount) and wait until the dice rolls in the favor of the adversary. Considering the ten seconds' block time, more than 3×10^6 reshuffles will occur each year, with the number of shards possibly reaching the hundreds, which provides an optimistic forecast for a patient adversary.

b) Bin sorting

To further impede the Sybil attack, the second step is added – sorting minters among shards in a way that assures a proportional stake distribution. Since the stated task is a variation of the partition and bin-sorting problems and hence is NP-complete, the optimal solution can create excessive computational overhead, which is why we use a simple approximate algorithm, as we do not need high precision.

All selected minters are sorted by the sizes of their stakes and split into bins according to the number of shards. Minters from each bin are then sorted by shards as the picture below shows, excluding the final round, when a greedy algorithm is applied to assure better optimization.

The algorithm forces the adversary to develop an adaptive strategy instead of simply splitting the stake among the highest possible number of Sybil accounts. It also solves the problem of uneven resources at stake, as simple random sampling would occasionally create shards in which minor stakeholders would form the Byzantine agreement supermajority.



Now let us investigate the most dangerous attack scenarios that could possibly be executed by a rational adversary.

a) **The system operates in the blockchain mode.**

When Dynemix operates in the blockchain mode, a single shard takeover does not pose a serious threat, since validity and consistency are assured during the master consensus; hence, the attack opens up only a censorship opportunity. Censorship starts posing a real threat when the adversary is capable of getting substantial representation in each block. Under such conditions, the adversary can censor third-party IMIN transactions, which can eventually allow the adversary to take over the entire system.

If the adversary intends to seize control over the system, the most obvious solution is to acquire the amount of resources close to the master consensus threshold ($\frac{3}{5}M$). Once the adversary controls about $\frac{3}{5} - \varepsilon$ of the global stake pool size, he or she will be able to control the master consensus and apply arbitrary censorship.

By violating fairness and liveness at the shard level, however, the adversary can develop a strategy that will allow success with many fewer resources.

Say the adversary possesses the amount of resources that allows him to continuously obtain a consensus threshold in a number of shards and a liveness threshold in a greater number of other shards, respectively. He censors the IMIN transactions of other stakeholders in those shards in which he obtained a consensus threshold, and blocks the consensus in those shards he obtained a liveness threshold. By doing so, he improves his relative representation in the next minters' committee. If this strategy is consistently executed, eventually he may have a chance to obtain a master consensus threshold, thereby seizing control over the system.

According to our approximate tests, in such a scenario, controlling about 30–35% of the global stake pool can provide an opportunity to succeed within a reasonable time frame.

b) **The system operates in the quilt mode.**

When Dynemix switches to the quilt mode, an eventual shard takeover can allow the adversary to breach validity, since the complete transaction data is not propagated globally. For this reason, a rational attacker does not need to execute complicated strategies and is more likely to concentrate on a simple shard takeover and an approval of a fraudulent transaction. This attack would require the possession of fewer resources and poses fewer risks for the adversary than the master consensus takeover attack described above.

According to our approximate tests, controlling only about 2% of the total stake would eventually allow the adversary to gain control of at least one shard for one round within a reasonable time interval.

ⓘ We should emphasize that the stated thresholds (2% and 30%, respectively) were obtained as results of approximate tests. The precise actual amount required for the attacks described will depend on the degree of the stakes' dispersion and the adversary's strategy, which is why further research on the subject is required for a more precise assessment.

In the quilt mode attack, resilience seems to be insufficient, considering the possible consequences of an attack, and additional measures are required, which include the following:

a) **Increasing the potential global stake pool**

The percentage of resources in the adversary's possession can have a totally different interpretation in absolute numbers, depending on the size of the global stake pool. When we evaluated Bitcoin's practical attack resilience, we stated that a 51% attack could be described as a 0.3% attack if we took into account the amount of required resources relative to the total market value of the platform at the time of the attack.

If we operate under an assumption of a 2% adversary's stake (which is very approximate), we can conclude that creating a global stake pool of the size equal to 15% of the circulating volume raises the quilt mode security guarantees to the level of Bitcoin.

We applied a number of measures to assure a larger global stake pool size (which at the same time helps raise the level of decentralization):

- Setting a small minimal stake amount

  Although lowering the minimal stake threshold allows the adversary to create more Sybil minters, at the same time it lowers the participation threshold for the general audience, which is expected to provide more security and decentralization, thus balancing out the negative side effect. In addition, the bin-sorting algorithm renders the mentioned Sybil strategy non-optimal.

- Securing low hardware and bandwidth requirements

  This helps common users participate in minting without excessive overhead, which, together with the previous measure, should lead to the involvement of more ordinary users.

- Allowing funds to be unbonded quickly

  Since the stake amount can be unblocked and spent almost instantly, participants are incentivized to use all available tokens as stakes, which greatly increases the size of the potential stake pool.

- Setting a high issuance rate

  As Dynemix features constant token emission, the participants can have expectations of token devaluation, which is why rational owners of large numbers of coins will likely try to participate in the minting process to compensate for inflation.

- Adopting the model of weak responsibility

  Allowing sporadic participation attracts additional stakeholders who cannot maintain constant availability.

For a more detailed description of the token issuance model, please refer to the Economics chapter.

Given the measures described, we expect the global pool to exceed the 15% margin, which means that Dynemix's quilt mode may potentially operate on the same level of security as Bitcoin or even surpass it (although we should note that further research is required to reliably confirm this claim).

b) Building the second line of defense

Although the measures described help strengthen security and presumably even raise it to the level of classic PoW blockchains, we still find the security guarantees that we can provide in the quilt mode insufficient.

As we intend to create a global digital currency payment system, more security is never superfluous.

During the development of Dynemix, we considered the following measures:

- Adding another round of consensus

  Since the problem of additional security can be solved by increasing the size of the shard committees, which we cannot afford due to the limitations of Guess My Block, we can simply add another BFT consensus round that involves the required number of participants.

  Say ten minters play Guess My Block and decide on a shard-block candidate. Upon receiving the BA result from the committee, another 500 randomly sampled notaries take a binary vote on the question of whether to approve or reject the shard-block candidate, thus confirming its validity. The second consensus round finalizes the shard-block.

  Although this approach allows the shard takeover threshold to increase greatly (pushing it to $\frac{2}{3} - \varepsilon$ of the global stake pool) by increasing the size of the validator sample to a scale that nearly nullifies the statistical dispersion, it negatively affects finality latency and throughput; a consensus round of hundreds of participants will take significant time to process due to the excessive communication overhead.

  Although there are some proposed solutions to reduce communication complexity for BFT algorithms, we do not find the inherent tradeoffs of these solutions acceptable. For example, using BLS threshold signatures can greatly reduce the amount of data exchanged between participants, but the concomitant increase in computation overhead can balance out the positive effect.

- Involving authorized notary (witness) nodes

  Instead of adding another BFT consensus, we can offer authorized nodes to testify that the block was seen and verified. This approach does not necessarily require an agreement to be reached, and there can be different models of this concept's implementation.

For example, we can use the reputation-based assignment of notaries, DPoS voting or any other method of choosing a set of trusted nodes that can vote for shard-blocks and/or master-blocks they consider valid. The more votes each block gets, the more reliably finalized it becomes (different designs can weaken the consistency guarantees of the system to eventual or probabilistic consistency).

There can be of various sets of rules on the notarization procedure, but this model has a fundamental intrinsic flaw: since a small group of predefined players can decide whether the entire system accepts a block, this creates incentives for cartel formation and censorship strategies.

- Involving fishermen

  If the previous solutions presume that the block is not accepted until it is signed by a certain number of verifiers, the fishermen concept presumes the opposite – the block is considered valid until a fisherman triggers a fraud alert.

  This approach ensures better decentralization, as fishermen cannot directly influence the finalization of valid blocks, but on the other hand, it inherits a dilemma that is hard to resolve. If we can manage to provide a reliable solution to the fisherman's dilemma, this concept seems to be the most optimal for our system.

7. Master-nodes for additional safety

Our solution for the second line of defense is based on the concept of authorized fishermen. It allows security assumptions to be greatly increased, while at the same time retaining the decentralization achieved in the quilt mode and not delaying finality.

1) Fishermen concept

According to the initial notion, fishermen are nodes that perform various kinds of spot checks of data chunks in search of faulty data. If a fisherman finds a fault, he or she propagates an alert through the network and gets a reward.

a) Unintentionally faulty data

Fishermen may be an effective solution to provide redundant fault tolerance, but blockchains do not need it, because blockchain technology features high common fault tolerance by default.

If we assume that a state change is replicated on a number of nodes that should get the same output in order to reach consensus, we can claim that sufficient redundancy is provided at the consensus round. There is a near-zero chance that a number of independent minters will simultaneously suffer the same common fault, which makes fishermen useless in this case.

b) Maliciously faulty data

Though there is clearly no point in the fishermen searching the blockchain for unintentionally faulty data, they might be interested in searching for maliciously corrupted data. If there is a possibility of a shard consensus takeover, there is also a chance of finding an invalid state change.

The problem is that the adversary will simply not give out the data. Suppose the adversary creates a fraudulent shard-block and the fisherman requests the transaction data. The adversary knows that if he or she discloses the data to the fisherman, the latter will certainly propagate it through the network and every node will be aware of the fraud. That is why the optimal adversarial strategy is to keep the data unavailable.

c) **Unavailable data**

If the adversary ignores the fisherman's request for the data, which is the optimal behavior under the assumption that the data is fraudulent, the fisherman will have no proof of adversarial behavior and will not be able to trigger an alert and receive a reward.

The only option is to allow fishermen to trigger an alert if the node ignores their requests, but this solution begets a dilemma.

2) Fisherman's dilemma

Since concealing data is the optimal strategy for the adversary, we can state that fishermen have no economic interest unless we provide an opportunity to act in the case of data non-availability. On the other hand, if we grant fishermen such authority, it creates preconditions for its abuse.

a) **Either fishermen or minters can abuse their opportunities.**

If we try to set rules for a rational environment, any model of economic incentives will feature a contradiction, as neither is the fisherman able to prove that the minter received the request and ignored it, nor can a minter prove that he or she submitted the requested data.

Suppose we granted fishermen permission to trigger an alert if minters do not satisfy the request. In this case, any fisherman can initiate a DDoS attack on their chosen minter at a near-zero cost. A malicious fisherman triggers an alert, and the entire network spams the minter with data requests. The minter cannot ignore it; otherwise, he will be accused of fraud. Nor can he prove that the fisherman's claims were deliberately false.

Suppose we added a kind of a stake requirement for fishermen, so that in the case that a fisherman triggers a number of false alerts, we can apply punitive measures on him. Then malicious minters will intentionally conceal valid data from fishermen in order to trigger alerts and present the data afterward, thus draining their stakes until no fishermen are left.

In addition to the problem of unbalanced incentives, there is a problem of near-zero success expectations.

b) **The fisherman has low motivation, knowing that he or she will not succeed because the adversary is aware of his or her presence.**

If we suppose that the fisherman and the adversary are players who try to think a step ahead of each other, their attempts to develop a winning strategy in their rivalry will face a contradiction.

The adversary knows that if he or she tries an attack, a fisherman will likely discover it and his or her resources will be wasted. This is why the adversary will likely refrain from attacking while fishermen are present in the system.

The fisherman understands this and concludes that there is no economic sense in suffering the waste of resources with a near-zero chance of succeeding against the adversary.

Under these assumptions, both the fisherman and the adversary find themselves playing a draw duel like cowboys in westerns. The only difference is that, in our case, none of the players is likely try to draw a gun due to expectations that his or her move will be effectively countered by the opponent, which means that the optimal solution is simply not to play.

Taking into account the above, it can be stated that fishermen can strengthen the security of the system by doing absolutely nothing – the mere threat of their presence forces the adversary to adjust his or her strategy.

On the other hand, there is always a possibility that some daredevil may commit an attack supposing that no fishermen are active due to the problem of low success expectations, which is why we do not consider this model sufficient for building security guarantees upon.

3) Major stakeholders as fishermen

Our solution is built upon an assumption that there is always a category comprising nodes that are indirectly incentivized to keep the system secure, which is why they can perform the functions of fishermen even without any direct economic incentives.

As we have explained, nodes can operate in either the quilt mode, which has lower hardware and bandwidth requirements, or the blockchain mode, which, not being fully affected by sharding, significantly raises the overhead. Nodes that keep the record of the entire blockchain are called master-nodes.

Suppose the adversary overruns the consensus in a shard and commits a fraudulent shard-block. Other minters, who operate in the quilt mode, obtained only the patch of the shard-block and, unable to verify whether all the transactions initiated the state change occurred fully according to the rules of the protocol, accept the patch as valid.

To verify the shard-block and make sure that it is valid, a node must obtain the entire shard-block from the adversary, but the adversary can ignore such requests, being aware of the consequences.

Suppose there are a number of fishermen who constantly search blocks for faults and are authorized to trigger an alert in the case of data non-availability. The adversary understands that he or she has little time to profit from his or her fraudulent behavior. The only option is to try to exchange his or her coins for an off-chain asset as quickly as possible, before the fraud is detected.

The adversary will likely turn to major players who can instantly accept a large transaction, which will include stock exchanges, market aggregators, providers of financial services etc. Given that such players are most likely to become victims of attacks and that they operate with significant token volumes, they have strong incentives to run master-nodes to assure better security for themselves.

At the same time, as such services aggregate large numbers of resources, they will likely have significant constant coin reserves. Considering that minting provides additional income and due to the fact that, in Dynemix, minters are allowed to unstake almost instantly, which means that staking most of the reserves will not deteriorate their liquidity, such services are incentivized to participate in minting as the largest stakeholders.

Combining these circumstances, we can conclude that the largest stakeholders will presumably run master-nodes, which will verify the entire blockchain and will be the most likely victims of fraud attempts. Under such assumptions, it seems logical to grant such players the authority to alert the network in the case of a fraud suspicion. Since there is still a small chance that even a major stakeholder may abuse his or her powers to harm arbitrary minters, an aggregate claim of three master-nodes is required for a full-scale alert.

Suppose that Alice, Bob, Carol and Dave run stock exchanges. Malicious Malory captures a consensus supermajority in a shard and adds a fraudulent transaction to the shard-block. This transaction transfers funds to Malory's account on Alice's stock exchange.

a) **Alice does not run a master-node.**

   Alice is careless enough not to run a master-node while dealing with large money flows. She accepts Malory's fraudulent transaction and commits a counter off-chain asset to Malory.

   Bob runs a master-node and requests data from Malory, but Malory does not respond. Bob suspects Malory of fraud and sends alerts to other authorized fishermen nodes.

   Bob presumes that if Malory is an adversary, she may likely have tried to transfer funds to a stock exchange (possibly Alice's). Even if Bob does not care about the safety of the system and the suspicious shard-block does not directly affect Bob's funds, he is not interested in letting Alice profit off of this situation, since Alice is Bob's competitor. Furthermore, causing economic damage to Alice is indirectly beneficial to Bob.

Carol and Dave, having come to the same conclusions, agree with Bob, and together they trigger an alert in the system.

After the network ensures that Malory's shard-block is fraudulent or unavailable, the shard-block gets banned, as do Alice's funds that were received from Malory. Malory gets away with the off-chain assets she received from Alice, and Alice is left with nothing.

This is why Alice should either run a master-node herself or at least wait until an alert time window ends, assuming that, if no alert was triggered by the master-nodes, Malory's shard-block must be valid. Considering that Alice runs financial services, she is interested in providing a quick response to her clients and keeping herself safe, which is why she will likely prefer to run a master-node of her own.

b) **Alice runs a master-node but does not trigger an alert.**

Alice runs a master-node and finds out that Malory's shard-block is unavailable, but she is too lazy to communicate with other master-nodes to trigger a global alert. She has already made her decision and does not accept the transaction from Malory, as she is suspicious that the shard-block is fraudulent. Since she has no incentive to trigger the alert and does not care about the safety of other users, she leaves the situation to the discretion of other participants.

In the case that all master-nodes suddenly turn lazy, they may find themselves in a fork, because, not being alerted, the network accepts the fraudulent shard-block as valid. This is why it is not reasonable to avoid their fishermen's responsibility if they count on keeping their data consistent with the entire network.

This example shows that major stakeholders are naturally interested in verifying blocks for their own protection, and we do not need to apply additional incentives for fishermen. At the same time, they are not expected to abuse their powers to DDoS minters, as it makes no economic sense for them and they are most interested in keeping the system operating stably. Moreover, the identities of major actors will likely be publicly known, and direct malicious behavior may cause reputational losses as well as various off-chain punitive measures applied to such actors.

Thereby, we circumvent the fisherman's dilemma and create a solution that can operate in a rational environment.

4) Master-nodes and decentralization

The presence of a limited set of nodes with special authorities raises a question of centralization. At first glance, it may seem that our solution reduces the decentralization level of the system, but in practice it does not.

Master-nodes cannot directly influence the state transition process but can only point out a possible fault. After a fisherman triggers an alert, the entire network investigates the situation and decides the fate of the specified shard-block.

The only threat that a malicious master-node can pose is the possibility of increasing the traffic load for a minting node by triggering an arbitrary false data non-availability alert. To accomplish it, the adversary needs enough tokens to simultaneously have three of his or her nodes among the largest stakeholders. Considering that Dynemix's design does not stimulate stakes aggregation (minting pools or similar aggregators), the adversary will likely need to obtain the required amount so that it is in his or her direct possession. Given the number of tokens needed to do this, it is hard to imagine a reason that a player of this magnitude would engage in such an activity.

In the case that numerous abuses are detected in the system, it is possible to adjust the rules and tighten the alert requirements, but we believe that this measure would be excessive, and the problem is unlikely to emerge.

In addition, the system can easily operate in the quilt mode even without a single master-node. The absence of master-nodes will lead to the following consequences:

a) **Security will be reduced to the level of quilt-layer guarantees.**

If none of the authorized fishermen actually run a master-node and perform fishermen functions, the adversary may succeed with a shard takeover attack.

The adversary, however, will not be able to distinguish whether any of the fishermen are actually performing data checks at any given time and adjust his or her strategy in advance. Given that any unsuccessful attempt will result in the loss of his or her Sybil accounts' stakes , we may state that the mere implementation of the fishermen concept already provides additional security, although we cannot formally define the degree of the security boost.

b) **History will be unavailable.**

Although no one restricts nodes from keeping the full blockchain history, there is no point in full nodes keeping data longer than proposed by the protocol specification.

As the system scales, the growth of storage overhead may become an obstacle to storing the entire blockchain, unless it is required for a particular purpose (for example, if the node is hosting a block explorer or a similar service).

c) **Data-availability guarantees will be reduced.**

If we assume that no one in the network accumulates shard-blocks, the transaction data will be obtainable only directly from the minters who built the block. Given the small number of participants in shard committees, there can be situations when the transaction data is temporarily unavailable.

This will not affect the opportunity to access the balances of any accounts or to send and receive payments, which means that the system will still work fine and users' funds will not be threatened. Without complete data, however, upon receiving funds users will not be able to identify the sender or amount of each transaction.

Taking into account that master-nodes cannot influence the state transition process and that their presence is not strictly necessary for the system's operation, we can state that the decentralization level reached in the quilt mode is not seriously affected by the presence of authorized master-nodes.

8. Overall security

1) Opposite attack strategies for different layers

The concept of authorized master-nodes brings with it a massive security boost due to the fact that breaking the master-node layer defense employs a totally different attack strategy than the one the adversary would need to execute to take over a shard consensus.

a) **A shard takeover strategy**

To approve a fraudulent shard-block, the adversary needs to control $> \frac{2}{3}m$ members of a shard committee. To accomplish this the stake should be split among a large number of Sybil accounts. Given the bin-sorting of the global appointed minters' committee among the shards, concentrating large stakes in a small number of Sybil accounts will not commonly allow the adversary to control more than 1–2 members of each shard committee, and the adversary will have to target different bins by placing stakes of different sizes.

b) **A master-nodes takeover strategy**

To breach the master-node-layer security, the adversary needs to control all except two authorized master-nodes. Given the 200 nodes limit, this would require controlling a stake of the size more than 198 times larger than the stake of the third largest stakeholder. This means

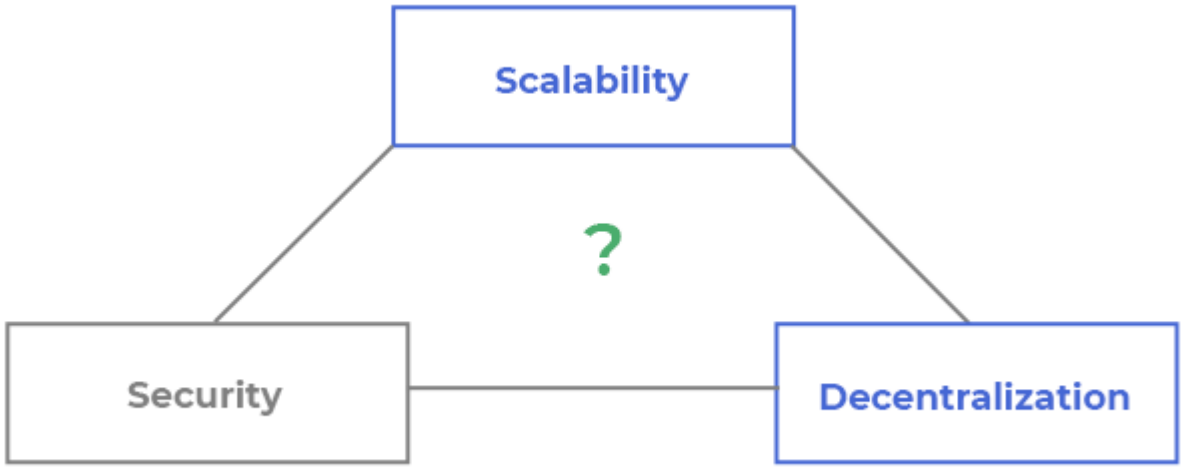
that the adversary needs to accumulate large stakes on a small number of accounts, which contradicts the shard-takeover strategy.

Combining the number of tokens needed to control the required number of master-nodes and at the same time having the chance to take over a shard consensus, we may presume that this model provides sufficient security. We cannot formally define the amount of resources needed for a successful attack, as it depends highly on the actual stakes dispersion, but intuitively we can presume that security guarantees will likely be more than sufficient.

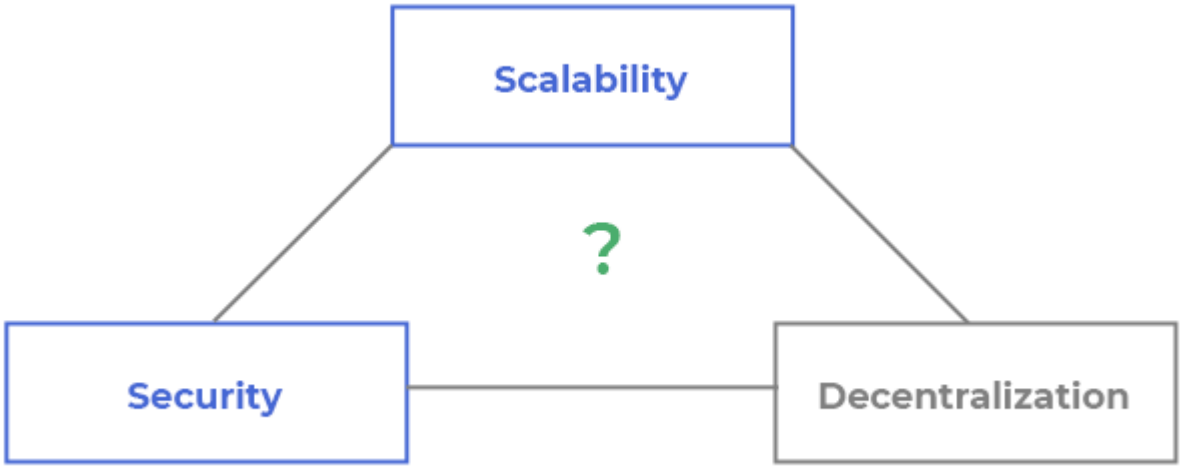
The master-node security layer can be also breached by a successful DDoS attack. A simultaneous DDoS of 198 professional nodes, however, which can also engage several instances run in different locations, is a highly unlikely scenario.

2) Merging two triangles

By merging two layers with different architecture, we have managed to provide a multi-layer solution to the scalability trilemma without the need to sacrifice either security or decentralization.

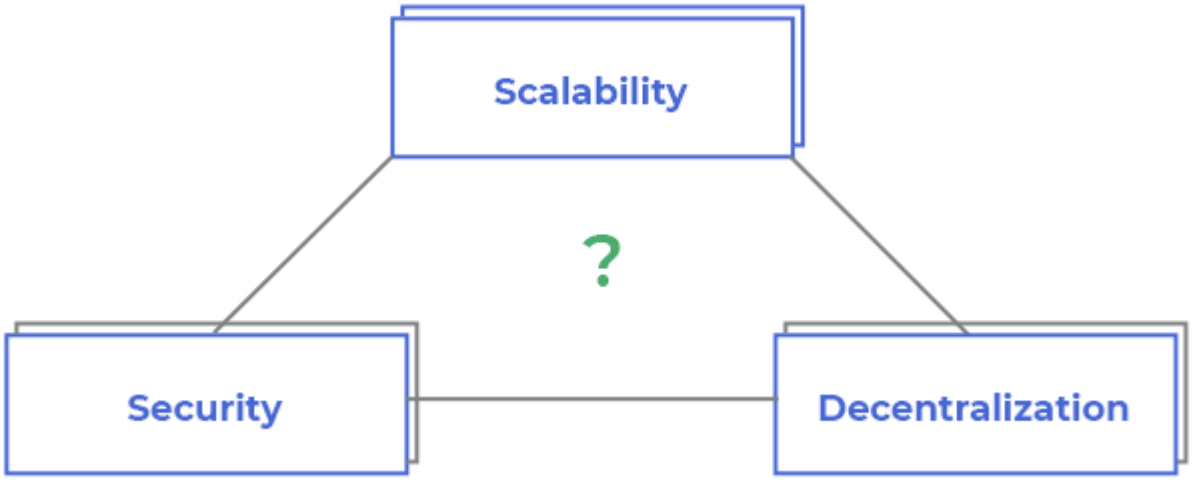


The quilt layer provides more decentralization, as it features much lower hardware and bandwidth requirements for nodes that operate on this layer. At the same time, security on this layer, though comparable to the security level of classic PoW blockchains, does not seem to be strong enough to satisfy our high standards. We may state that the quilt layer features high scalability and decentralization but lacks security.



The blockchain layer, being not fully affected by sharding, lacks decentralization, for as the system scales up, the computation, storage and traffic overhead grow linearly. At the same time, it provides strong security, as the adversary would need a large number of resources to gain control of the

fishermen nodes. That is why, on the blockchain layer, decentralization is sacrificed for the sake of security.



Being combined, two layers compensate for each other's shortcomings, at the same time keeping the advantages of each layer intact. The master-node layer provides additional security for the quilt layer but does not reduce the overall decentralization achieved.

9. Achieved scalability and its further possible increase

1) Efficiency of the current design

The main purpose of sharding is to distribute the computation and communication overhead of peer nodes, allowing the system to process more transactions with the same hardware and bandwidth assumptions.

If our sharding solution features the global state, which is updated on all nodes, it raises the question: how does it reduce the overhead if it still requires propagation of the same amount of data as classic non-sharded blockchains?

To begin with, our solution allows communication overhead to be optimized for participating replicas and at the same time to provide more DoS resilience by significantly increasing the total number of participants.

Another important matter is that our sharding scheme allows to solve several issues that emerge from other features of the protocol design. Namely, it helps with the proper implementation of Guess My Block game and the reward distribution scheme.

Finally, the protocol was initially designed to operate in the quilt mode, which provides a significant performance boost, especially after the implementation of homomorphic encryption. At the same time, the system can be initiated with all nodes operating in the blockchain mode (hence being master-nodes), and smoothly transition to quilt and the subsequent encryption implementation without the need for significant architecture reconsideration.

We find it dangerous to start directly in quilt mode, due to the safety issues described above. During its infancy, the system should operate in the blockchain mode, which provides stronger validity guarantees and features a higher adversarial threshold without engaging authorized fishermen master-nodes. After the potential for scalability in this mode is capped, the system can be switched to the quilt mode, which will further increase its scalability. Finally, when the appropriate balance encryption scheme is developed, it can be painlessly implemented into the system, which will already be operating in the quilt mode.

As the protocol allows subsequently switching between modes of operation, we can model the stages of development and estimate how the system will scale over time.

In our tests and calculations, we modeled the computation and communication capabilities that most users can meet using affordable hardware. Our particular setup is as follows:

- 10-megabit bandwidth;
- 300 ms average latency;
- Intel Core i5 7360U (a mid-range outdated laptop CPU);
- Mid-range SSD.

We assume that users should not experience a 100% hardware load, as it would be inconvenient, which is why we limit the use of the CPU to a single core.

a) **The blockchain mode (or the master-nodes mode)**

Dynemix blockchain is a chain of master-blocks that are assembled from a set of shard-blocks. Shard-blocks contain transactions themselves as long as a lot of metadata, which is why the complete master-blocks take significant time to be transmitted and validated.

Given the described setup, the system will be able to process approximately 600 TPS using a single CPU core for computation.

b) **The quilt mode (or full nodes mode) without encryption**

This mode operates with the state of the entire system, which is updated by quilts – data structures that contain only a part of the information of the blockchain blocks. Full nodes obtain the current global state and then only apply quilts to execute state transitions. A quilt can be derived from the master-block, but the block cannot be restored from the quilt.

Full nodes experience greatly reduced computation, traffic and storage overhead, as they do not need to process transactions (except while directly assembling shard-blocks).

Given the setup described, the system will be able to process approximately 2,400 TPS using a single CPU core for computation.

c) **The quilt mode (or full nodes mode) with encryption**

Homomorphic encryption with ZK-proofs for each transaction will severely increase both computation and communication overhead, and the only option for providing a scalable solution is switching to the quilt mode. The problem is that, currently, we cannot predict the actual parameters, but we can assume that the more the overhead increases, the more efficient the quilt solution will become.

We can clearly see that the system is capable of providing decent scalability even with very modest setup assumptions. 2,400 TPS is an enormous number, which can be capped only when the system gains worldwide popularity. We assume that by that time we will be able to significantly raise hardware and bandwidth assumptions, which means that Dynemix will be unlikely to suffer from insufficient scalability.

Let us upgrade our setup to the level of current top-range consumer specs:

- 100-megabit bandwidth;
- 300 ms average latency;
- Intel Core i7 9700 (a current gen top-middle desktop CPU);
- Fast SSD.

In this case, we assume that the user can share the entire bandwidth and CPU power, taking into account that in 5–10 years the same level of performance will likely be achievable with only a partial load on average consumer-class hardware. Under such assumptions, performance will rise approximately to the following numbers:

- 4,000 TPS in the blockchain mode;
- 16,000 TPS in the quilt mode.

Given the specified results, we believe that the current design will provide sufficient scalability at any stage of the system's development, while at the same time keeping hardware requirements tolerable for most consumers around the world, thus maintaining the desired level of decentralization.

For this reason, we did not go for a complete sharding scheme – we simply do not see the necessity to push scalability further, and we find it more reasonable to employ a more robust and efficient partial sharding design.

We still cannot build any reliable precise assumptions about performance, however, in the case of the implementation of homomorphic encryption. We concede that an actual encryption scheme could require a much higher overhead than expected, and it may happen so that we will face the necessity of pushing scalability further by adjusting the design and accepting other tradeoffs.

2) Splitting state into active and archive

With the growth of the userbase, the storage overhead may become the first bottleneck of the system. A billion users' states would weight about 120 GB raw and 540 GB in an encrypted form, which likely exceeds the amount of storage that an average non-professional minter is ready to share. Synchronizing a state of that size would also be a serious challenge for most users.

One possible solution to reducing storage overhead is to shard and archive the state of inactive accounts. Considering the statistics, we may expect that many addresses may be inactive for long periods and that only a minority of users will issue transactions at least monthly.

Under such assumptions, it is reasonable to split the system state into different layers, which will then be stored and accessed according to different rules:

a) **Active state**

This layer includes everything needed for block production (stakes, IMINs) and states of accounts that were active at least once within a specific period of time (e.g. last month). If an account doesn't change its state in the determined time interval, its state converts to "archive."

b) **Archive state**

After the account stays idle for a month, its state is archived and then deleted by most nodes. The archive is sharded and stored on a redundant number of nodes, from which it can be easily recovered whenever necessary. As user clients also store states along with cryptographic proofs of their validity, when an archived account issues a transaction, the client includes the state, and the transaction is processed without any additional delay.

These measures can significantly lower the storage overhead at the cost of a slight increase in traffic overhead, which is overall an acceptable tradeoff.

3) Implementing complete sharding (without global state)

This is a more challenging goal. Though we could keep many elements of the current design, it would still require a serious architecture reconsideration, as new solutions for several emerging problems need to be developed.

Fortunately, certain features of the current design can significantly help build an appropriate sharding solution with isolated states. In particular, the Guess My Block game can reliably ensure that both parts of a cross-shard transaction will be processed instantly and simultaneously, which means that we can provide a solution to the train-and-hotel problem that will fit our finality latency standards: namely, we can use a two-phase consensus to finalize a cross-shard transaction.

We will still have to accept certain tradeoffs, however, that will inevitably degrade some properties of the system. We find the current design almost perfectly balanced from the point of view of targeted properties and setting, which is why we consider complete sharding the last resort for the scalability issue.

Furthermore, it is not currently clear whether we will need to push scalability this far and whether a complete sharding solution will actually provide a significant scalability boost that would be worth the concomitant tradeoffs.

10. Partitioning and forking

1) Partitioning and CAP theorem

From the point of view of CAP theorem, Dynemix is consistency-focused. We can conditionally call it a CP system, but according to the initial notions of CAP theorem, it is more appropriate to call it “C and moderately P.”

The problem is that Dynemix cannot fully tolerate *arbitrary partitioning*. As the system operates according to *the weakly synchronous* model, liveness holds only as long as a required threshold of participating replicas is allocated in the same partition.

In the case that a node is not able to get enough responses from the other replicas by the end of the subslot, it assumes that it has been isolated in a partition and locks in *the recovery mode*, whereby it waits until a valid master-block is seen. Upon restoring connectivity and receiving the current master-block, the node restores consistency with the rest of the network and rejoins the protocol.

It is easy to conclude that if the system splits into a number of partitions, none of which contain enough replicas to assemble at least a single shard-block and then reach a master consensus ($\frac{3}{5}M$), all replicas will enter the recovery mode and the protocol will stall.

Given the assumed network topology of Dynemix, we consider this highly unlikely. A practical partitioning threat refers mostly to situations in which a small fraction of nodes get isolated from the major network segment, which should be handled fine by the Dynemix protocol.

Nevertheless, what if this actually happens and the system stalls due to the violation of synchrony? We assume that this is an extraordinary situation, which indicates a global cataclysm. In this case, it is reasonable to let the system be recovered in manual mode after the consequences are overcome.

We can theoretically propose an automatic fallback recovery algorithm, but the problem is that by doing so we will inevitably extend the communication model to asynchrony or partial synchrony, with all their respective consequences. We would likely no longer be able to guarantee either adaptive corruption tolerance even against a mildly adaptive adversary or sequential consistency, which seems like a heavy price to pay.

We have chosen consistency over availability for the following reasons:

a) **We believe that consistency is preferable for a consumer-level payment system.**

   If we assume that the system will be used by a general audience, we cannot go for availability and allow weak (eventual or probabilistic) consistency. This results in the possibility of forks and complicates the notion of finality (as we noted, fast finality is one of the crucial features we need to achieve).

Common users perceive a payment system as something solid and reliable. This means that if the transaction is processed, no rollback is expected to occur. If the user is isolated in a partition, it is obviously better not to provide him or her with an incorrect state and not to accept his or her transactions than to allow them to be rolled back after the system recovers.

Availability may be an option for specific blockchains focused on professional use, when we assume that users understand the consequences of inconsistent system operation, but this is clearly not our case.

b) **Homomorphic encryption does not allow forks to be merged.**

If balances are not encrypted and the system suffers from partitioning, it is possible to adopt merging rules that are applied when the network recovers. Transactions from one fork can be relocated to the other fork (unless double spends occurred), and users will not suffer a rollback when the lesser chain is orphaned. If balances are encrypted but the system uses a UTXO model, relocation of transactions is also possible.

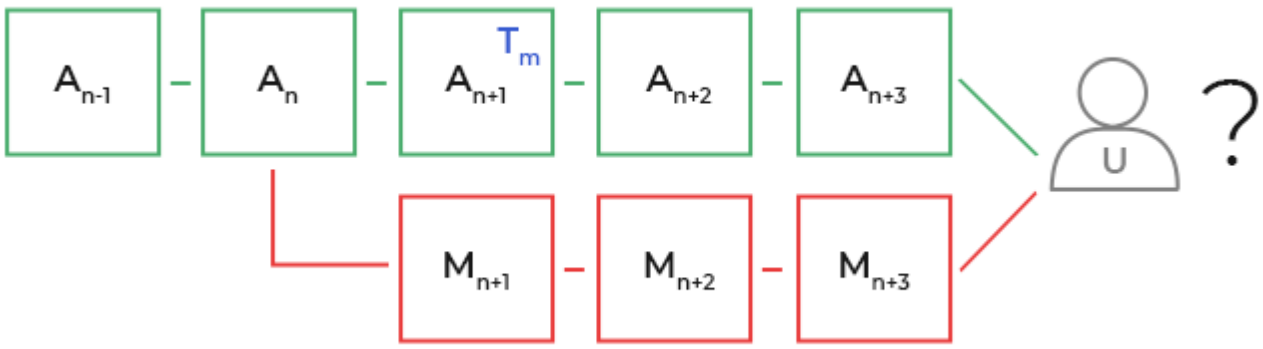
A homomorphically encrypted account-based state, however, does not allow fork merging. The reason is that ZK-proofs become invalid if the state changes. This is why if a user receives funds in partition A, his or her transactions in partition B will become invalid for partition A.

2) Forks

By being consistency-focused and featuring a BFT-style consensus model, Dynemix is designed to strongly maintain a linear blockchain structure, which means that no forks can occur in the system unless an attack is committed.

All shard committees for block B_n are formed according to the information included in block B_{n-1}. A master-block is considered valid if at least $\frac{3}{5}M$ committee members voted for it. This means that two valid blocks of one height can appear only if a number of minters sign both of them. This cannot happen unintentionally, and it clearly indicates the malicious behavior of such minters.

Since we can be sure that each fork appearing in the system signifies an attack, we can apply punitive measures to the participants in the alternative fork.



Assume that chain A is the main fork of the network that is minted by honest minters and that Malory is an adversary who wants to commit a double spend. To accomplish this, Malory issues transaction T_M, which transfers her funds to a stock exchange. Minters include the transaction in block A_{n+1} , after which Malory exchanges her funds into an off-chain asset. Now Malory wants the network to switch to an alternative chain in which T_M never happened and her funds stayed in her account (or were transferred somewhere else).

She chooses an arbitrary branching point, which obviously should occur before block A_{n+1}. Suppose she chooses block A_n as the parent and starts alternative chain M from height $n+1$. After she completes all the preparations, she reveals her fork to the network. Suppose it happens on height $n+3$. Now all nodes in the network must decide which chain they are going to join. The decision is made according to the following rules:

a) **Nodes that witnessed transition $A_n \rightarrow A_{n+1}$ will not buy the bait.**

All nodes that are constantly online and keep the record of the entire blockchain (at least the hashes of blocks) know that block A_n was followed by block A_{n+1}. At the time block A_{n+1} was propagated, block M_{n+1} was not observed in the network and hence must have been revealed later, which is why the nodes consider chain A valid and chain M adversarial. They inspect the latter and slash the stakes of minters who participated in that chain (if any of them have not unstaked).

b) **Nodes that witnessed any block in chain A since A_{n+1} will not buy the bait either.**

If the node did not witness the transition but is aware of any block on chain A between the branching point (height $n+1$) and the current block (height $n+3$), it comes to the same conclusions as in the previous case. For example, if the node is aware of block A_{n+2} and did not see any alternative blocks at that height, it means that A must be the valid chain.

c) **Newcomers or nodes that did not synchronize with the network since A_n will have to choose a fork.**

If the node was last online before the branching point or if it simply has just joined the network, it cannot know which chain was minted secretly by the adversary and which one was built by honest minters.

3) Chain selection for non-witnesses

At this point, we address the problems of *subjectivity* and *nothing at stake*. As we stated in the chapter devoted to comparing PoW and PoS, PoW is considered objective, for any node can choose the correct fork without the need to trust third parties, while PoS is inherently vulnerable to posterior corruption and a newcomer may be unable to distinguish the adversarial chain from the honest chain relying on the protocol rules exclusively, which is why PoS protocols are subjective.

We do not support this division, however, as we do not believe that overall objectivity is practically achievable with the help of any blockchain design. Despite the fact that Bitcoin may be objective on the protocol layer, in practice user–protocol interaction necessarily includes intermediate layers (e.g. client, operating system, hardware etc.), which cannot satisfy the same objectivity conditions. This is why we consider any blockchain implementation in practice subjective to the same extent.

If we assume that a user can verify a Bitcoin blockchain only with the help of software and hence needs to trust the provided result and presume that the software was not corrupted, providing the current state of the blockchain along with the software does make it any more subjective.

For this reason, the problem of chain selection for newcomers can be solved trivially – the hash of a recent master-block should be provided along with the client, or a set of trusted bootstrapping nodes should be predefined for initial synchronization. We do not believe that any more sophisticated solution is required, as subjectivity cannot be circumvented no matter how complicated an algorithm we propose, and we see absolutely nothing wrong with that.

The only category left unaddressed so far is that of nodes that were last online before the branching point and did not witness any blocks above n height.

The main problem with proposing a secure algorithm for this case is *nothing at stake*. While the work applied to a block constructed according to PoW design cannot be transitioned to an alternative block of the same height, in PoS protocols the same stakes can be used to construct an arbitrary number of parallel forks at a near-zero cost, which makes them vulnerable to posterior corruption. A possible solution to the problem was described by Vitalik Buterin and denoted as a weakly subjective model. In short, the solution looks like this:

- Adopt a slashing algorithm, which penalizes stakeholders who behave incorrectly (for example, vote for more than a single block at one height);

- Lock stakes as security deposits for N blocks to prevent stakeholders from reverting history starting from a branching point less than N blocks deep without the threat of losing their stakes;
- Set checkpoints at the height at which stakes are unlocked, beyond which no alternative forks are taken into consideration.

This means that if the node was last online fewer than N blocks ago, the blockchain keeps roughly the same level of security as PoW blockchains. If the node was unavailable for more than N blocks, the protocol cannot guarantee secure chain selection for it anymore.

According to the initial idea, N should be set to cover a large time interval, possibly reaching months to be effective, which means that stakeholders have to freeze their deposits for a long time.

Although this solution partly solves the nothing at stake problem (at least for N range), we do not believe that such a design is optimal for Dynemix for the following reasons:

a) **Locking stakes for a long period severely deteriorates several properties of the system.**

The main argument against long-term deposits is a concomitant dramatic change in the quality and volume of the global stake pool.

One very important achievement of the Dynemix design is the involvement of ordinary users in the support of the system. Not only did we manage to lower the hardware requirements for minting nodes to the level of consumer hardware, but we also built an economic model that does not create any obstacles to users' becoming stakeholders.

Unlike most blockchains, in which block constructing tends to be a more professional activity with a high participation threshold, Dynemix is designed to be a system in which an average user can participate on equal footing with professional block producers. This brings Dynemix much closer to being a true peer system and the embodiment of the initial ideas behind cryptocurrency technology.

The ability to quickly unbond and spend a user's funds is an essential feature that helps achieve this goal. If we deprive users of this opportunity, this will instantly rearrange the stake pool in the favor of professional participants. The decision to freeze funds for half a year requires serious consideration and is obviously not something an average user can easily afford.

Moreover, this will also affect major stakeholders who provide financial services and need liquidity. If we allow a quick unbond, stock exchanges and similar actors can stake most of their reserve without any threat to their activities. Long-term deposits, however, can deprive them of the necessary liquidity.

This shows that implementing long-term deposits will almost certainly decrease decentralization and security both for full nodes and master-nodes. This is an unacceptable tradeoff, especially considering a highly disputable positive effect of this solution.

b) **The proposed solution helps only a small fraction of users.**

The weak subjectivity model may be relevant to blockchains that allow forks (featuring a common prefix design and the longest chain rule), but Dynemix is strongly consistent, and no forks are supposed to occur (unless a very powerful adversary commits a forking attack).

For this reason, a potential attack can affect only a small share of users who satisfy both of the following conditions:

- They were last online before the branching point;
- They have at least one of the adversary's nodes in the bootstrapping list.

In the case of an attack, the model described can only provide more protection for this category and should not generate any notable boost to the entire system's security, as most nodes will stay completely immune to attack attempts.

Furthermore, the affected nodes will only be subjected to a delay caused by the necessity of choosing the correct chain. Considering that these nodes will presumably need some time to synchronize with the network in any case, this does not seem to be an outstanding achievement for the adversary.

In addition, the weak subjectivity model provides no solution if the branching point occurred more than N blocks into the history, which means that the adversary can simply prepare for the attack and commit it after N passes. In this scenario, the attack will likely affect many fewer users than in the case of a more rushing attack, but at the scale of the entire system, the difference may be barely notable.

Given the stated arguments, we see no reason to implement the proposed model in its initial form. Instead, we adapted the principles of the model to the design of Dynemix.

The deposit lock period is set to two blocks. This allows users to quickly unstake and provide the necessary liquidity for all categories of minters. Such a short deposit period protects them only from double signing any data before each block is propagated and settled, since it provides just enough time to penalize Byzantine actors but does not solve the nothing at stake problem in terms of posterior corruption.

If the node encounters two conflicting blocks at one height that descend from an arbitrary branching point in history, given that the node has not witnessed any blocks after the branching point, the following resolution algorithm is proposed:

a) **Figure out whether both forks are valid.**

The node finds a branching point and requests a set of IMIN transactions with cryptographic proofs from the common parent master-block (A_n) and the signatures of the minters of the first divergent master-block (A_{n+1} and M_{n+1}) from each chain. According to IMIN transactions and the hash of A_n, the node establishes the shard committee's designation for $n+1$ height. The node verifies that both A_{n+1} and M_{n+1} were built according to the rules: i.e. both were signed by the required threshold of minters.

On this stage, the node needs to download only a small portion of data from three blocks, which is affordable for common users even after the system scales up (approximately less than 5 MB of data for a 100 million userbase).

If block M_{n+1} does not meet the requirements, the decision is instantly made in favor of chain A. If Malory has managed to obtain enough signatures to construct a valid M_{n+1}, however, the node proceeds to the next step.

b) **Figure out the preferences of trusted nodes.**

If Malory presented a valid fork, this already means that she has put a lot of effort into the attack, considering the number of minters' signatures she has managed to obtain (or control initially). On that point it is reasonable to bring in subjectivity and to offer the user decide him- or herself.

To help the user with the choice, the client investigates which fork is supported by nodes that are considered to be run by powerful actors who are unlikely to engage in fraud. This may include stock exchanges, developers, block explorers etc. This may also include nodes that gained a certain reputation from the point of view of the personal experience of the user's node. There can be different approaches to the construction of such a list, and no particular rules are needed on the protocol level.

The information collected should be sufficient for the user to figure out which fork is legitimate. At the same time, this kind of research will not consume any noticeable amount of traffic and can be performed in a couple of seconds regardless of the current system load.

Some users, however, may prefer investigating both chains more precisely.

c) **Download data from both chains and estimate which of them contains more voting power.**

If the user for some reason cannot make a decision in the previous step and wants to discover how much stakes each fork accumulates, he or she may download data from both chains and conduct research in the following order:

- The node establishes the total stakes in the system and builds the list of stakeholders at height n (the accuracy depends on the depth of the research).
- The node downloads data from each chain block by block and discovers the votes of the stakeholders. Service transactions (i.e. stake, unstake and IMIN) refer to the parent block and thus are TaPoS-compatible. They cannot be replicated by the adversary without re-signing, which is why whenever the node finds such a transaction issued by a stakeholder, it is considered a vote for a particular chain. If a stakeholder is found placing transactions in both forks, the node concludes that this account is compromised and does not count its votes.

By the end of the investigation, the node will have the following information (the accuracy depends on the depth of the research of the stakes at height n and the number of verified blocks from both concurrent chains):

- the number of accounts that voted for each chain;
- the number of stakes accumulated in each chain;
- the current amount at stake (funds that were not unstaked and that Malory actually risks losing).

This information will provide a comprehensive picture of the forks and allow the user to make a choice. The ratio of certain parameters may be also set for non-interactive selection. For example, if the client concludes that more than 99% of the stakes voted for chain A, this obviously must be the valid chain, as Malory is extremely unlikely to be able to collect that many stakeholders' signatures, even if the branching point occurred far back in history, which means that the client is able to make a reliable choice without user involvement.

The last step, however, is relevant only to the period when the system load is relatively low and the investigation will require a tolerable traffic overhead. If the system scales up to a 100 million userbase, downloading the complete data from both chains may seem unreasonable for the average user. If Malory sets a branching point a month deep and builds a chain up to the current height, the data required for the complete investigation will likely exceed a terabyte.

For this reason, we believe that, given the properties of the Dynemix protocol, we should not necessarily concentrate on a formal approach to the chain selection rules. Subjective selection may be a preferable option for most users, who will likely prefer to make a choice according to the information on trusted players' preferences rather than download and process terabytes of data.

Bringing a bit more subjectivity to the table also favors security. If we apply a formal approach, Malory will know particular conditions that will allow her to succeed. Transitioning chain selection rules from the protocol layer to the client layer, however, makes Malory's perspective uncertain. She cannot reliably predict whether a single user will fall for her trap, and, if someone does, whether she will be able to take advantage of it. Although there can be nothing at stake, at the same time there can be nothing in the guaranteed expected return.

11. Finality

Earlier in this chapter, we addressed the issue of finality, stating that it should be achieved as quickly as possible (at worst, within 20 seconds after the transaction is issued). At the same time, we did not explain what finality exactly means in relation to Dynemix.

The notion of finality differs for blockchain systems of different consensus design.

In Bitcoin and other systems that feature probabilistic consistency, finality increases exponentially with each new block mined on top of the block that contains the transaction. The degree of relative finality can be considered individually depending mostly on the importance of the transaction to the recipient. If the transaction features low value, one may consider a couple of blocks sufficient. In the case of a valuable transaction, it may be reasonable to wait for 6–10 confirmations.

In the case described, reaching finality is a prolonged procedure. It is a fundamental boundary of any system based on the common prefix design, which is why we initially rejected this architecture for Dynemix.

In consistency-oriented blockchains with a BFT model of consensus, finality is reached after nodes achieve a consensus or other special events occur, and in many systems, it depends on the number of approved blocks on top to a much lesser or even zero extent.

In Dynemix, we can outline the three main checkpoints of finality:

a) **Shard-block commitment (approximately 4–5 seconds after the transaction is received by a minter)**

   After the members of a shard committee reach a Byzantine agreement, the shard-block may be considered finalized under the assumptions that both shard- and master BA supermajorities are honest and the system's liveness is not breached.

   This means that the transaction may be reverted only in tree cases:

   - The system is under an attack that requires controlling the BA supermajority within the shard (a shard takeover attack);
   - The system is suffering an attack on liveness or partitioning due to a network malfunction.
   - The master consensus supermajority is corrupt and intends to ignore the shard-block.

   Since neither event is expected to occur under normal conditions, this checkpoint provides finality guarantees strong enough for most transactions, except ones of very high value.

   Since shard-blocks (or patches) are exchanged only among the minters of the current block, however, most users will not have access to them and hence will have to wait for the next checkpoint to get the confirmation.

b) **Master-block approval (approximately 8–9 seconds after the transaction is received by a minter)**

   After a master consensus is reached, the master-block cannot be reverted, which means that, upon receiving the master-block, the user can consider the transaction finalized with only a negligible probability of reversal.

   If there is a fraudulent shard-block approval by the corrupted consensus supermajority, however, the shard-block that contains the transaction can be banned, and the transaction can be reverted, which means that in some cases users may prefer to wait until the final checkpoint is reached.

c) **Block validation by master-nodes (approximately 18–19 seconds after the transaction is received by a minter)**

If the user deals with high value transfers, it is reasonable for him to run a master-node and verify shard-blocks himself. In this case, the transaction may be considered fully finalized when the user verifies the shard-block that contains the transaction and concludes that it has been assembled according to the rules of the protocol.

If the user does not run a master-node, he or she should wait until the next block is committed. By that time, the previous block should be verified by the master-nodes, and in the case that no alarm was triggered, the block must be valid, and the node may consider it fully finalized.

After the last checkpoint is reached, the transaction cannot be reverted unless a massive attack is committed that can fully breach all levels of the system's security. Additional blocks minted on top of the block that contains the transaction do not grant any stronger finality guarantees.

VI. The Liberdyne Messenger

1. Decentralized architecture

Liberdyne is a decentralized P2P messenger with an emphasis on security and privacy.

Liberdyne uses the account base and transport protocols of the Dynemix blockchain system, which greatly improve the messenger's capabilities.

Liberdyne is designed to withstand a significant load created by a userbase of a size comparable to that of popular centralized solutions.

1) Liberdyne and Dynemix blockchain

Unlike the typical approach, we did not aim to use technology for the sake of technology. Blockchain is used only to the extent that helps build a better system and improve the user experience. The irony is that we cannot call Liberdyne a blockchain messenger, considering the actual degree of blockchain involvement in the messenger's architecture.

There are several existing blockchain messenger projects, but blockchain technology itself is not optimal for storing and transferring data of low value, such as ordinary correspondence. Any messenger that stores transferred data directly in the blockchain is a very niche product that cannot compete with conventional client-server apps from the point of view of efficiency and scalability.

Liberdyne is closer to more common P2P messengers (e.g. Tox) in its design, but the Dynemix blockchain still plays an important role in providing a solution to several problems emerging from P2P architecture.

a) **Accounts**

Secure communication requires the ability to reliably identify interlocutors, which is why in a decentralized peering system, management of the account base can become a serious challenge. Blockchain technology offers an optimal solution providing secure decentralized consistent key storage, which is highly resistant to MITM attacks.

b) **Rewards for system support**

When it comes to system support, most P2P systems rely on altruistic behavior by the participants, who share hardware resources for the sake of community. This model may work to a certain extent, but in the case of a significant hardware load, altruistic intentions may not suffice. Dynemix helps Liberdyne build a fair system of economic incentives, which ensures smooth operation.

c) **Dynemix wallet**

An embedded cryptowallet is a quite standard feature of any blockchain messenger and often the only reason a messenger is actually called as such. Liberdyne allows payments to be made from Dynemix accounts or unregistered key pairs, which can be added if the user wishes to engage multiple wallets.

d) **Transport protocols**

Liberdyne uses the low-level transport protocols of Dynemix. This helps build the mentioned reward distribution system with the help of blockchain transactions and also provides for traffic obfuscation for user-generated content, making meta-data analysis significantly more difficult.

2. Accounts in Liberdyne

Liberdyne employs Dynemix accounts for its operation. To create an account in Dynemix, a master key should be generated that is then secretly stored by the user and engaged to log in. The Dynemix protocol has no requirements for key-generation methods; the only thing that matters is format matching.

There can be different approaches to key generation, however, which significantly affects the user experience. Since Liberdyne is a complete product, it should feature particular methods to meet the needs of most users.

During the account-creation process, users will be offered two different methods of key generation, each of which will be more suitable for certain types of system usage.

1) Random generation

Security: high | Convinience: low

For users who will store significant amounts of dynes in the account or transfer important information

If the user chooses this option, a master key is generated randomly. This approach makes the key uncrackable, but requires storing the key and entering it on each logon. Since a 256-bit randomly generated key is unmemorizable, users will have to engage some kind of a storage (from a simple sheet of paper to a special hardware wallet). Although a random generated key is immune to brute-force attacks, the wrong choice of storage may compromise the key itself and cause the loss of the account.

This method is standard for most blockchain systems, but since we are creating a product for mass audience, many common users may find this approach inconvenient.

2) Password hashing

Security: average | Convinience: high

For users who will store small numbers of dynes in their accounts or use the application for everyday communication

To improve the user experience, another approach will be offered – hashing a master key from a user-generated password. Though we will use a secure key derivation function (Argon2 or any other stronger solution that may be developed in future), this method is still vulnerable to brute-force attacks if the password set by the user is not strong enough.

The positive side is that the user will be able to remember the password (or the passphrase) and interact with the system in a much more convenient way. Since this method is familiar to most users, we believe that it will prevail.

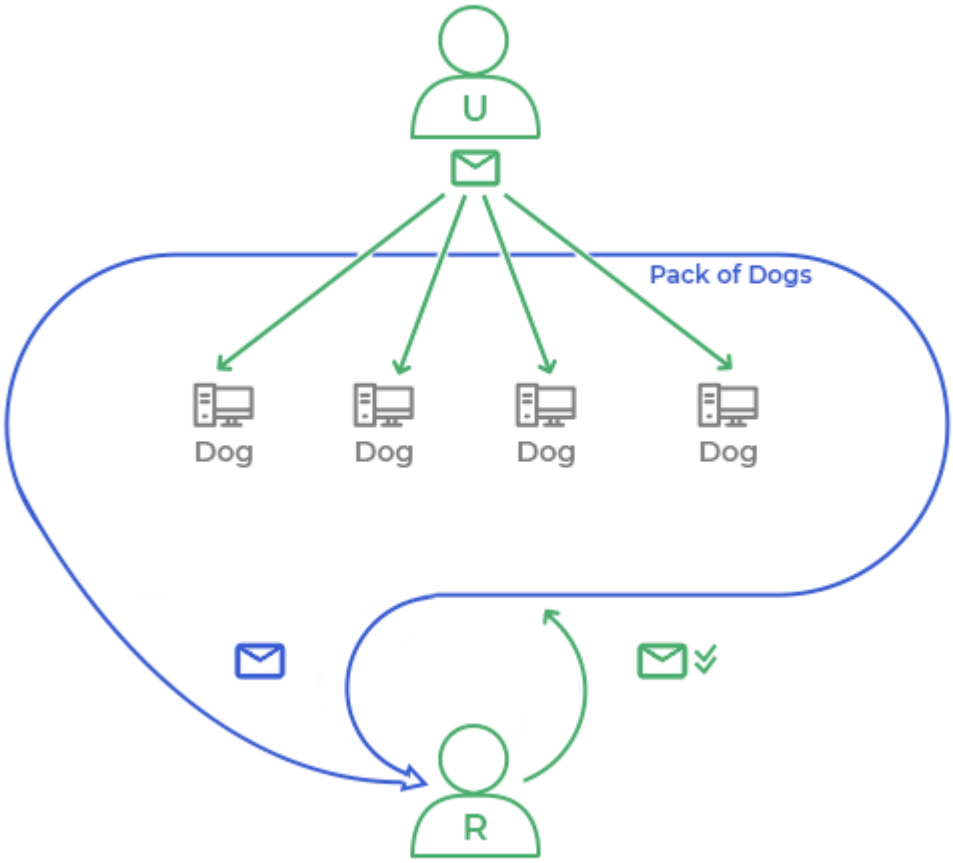
3. Delivery to offline guys (DOG)

P2P architecture raises a serious problem – delivery of content when direct P2P interaction is impossible. If the recipient is offline, the message cannot be delivered. The sender's client may try to connect intermittently and deliver the message when the recipient reconnects to the system, but what if the sender himself goes offline?

In centralized messengers, servers are used as delivery relays, so the message is stored on a server, which obviously never goes offline until the recipient connects to the network, and thus the problem is solved. In a P2P network, peers perform the relay functions, which is why such a solution will not work.

To solve this problem, we implemented a special protocol of delivery to offline guys (or girls, if you prefer) – DOG.

1) Delivery protocol



If user U tries to send a message to recipient R but fails to connect to the node of R, he or she finds several DOG nodes (or simply *dogs*) and commits the delivery to them, thus forming a *pack*. U

informs each dog about all other members of the pack so that they will be able to maintain the size of the pack.

If U goes offline, the replicated messages will still be stored on the packs' nodes, each of which will intermittently try to deliver the message and check whether the entire pack is still online. If one or more nodes of the pack goes offline, the remaining dogs will find substituting nodes, thus keeping the pack size constant.

When R finally connects to the network, one of the dogs delivers the message. The others will attempt the delivery as well, but R will respond that he or she has already received the message.

After all dogs get delivery confirmation, the pack is considered disbanded.

It is worth noting that DOG is not strictly a Liberdyne feature but a low-level transport protocol of the Dynemix system, which means that any type of service data can be delivered via DOG if needed.

2) Rewards for dogs

Dogs will be rewarded with newly issued coins. This is a feature that makes Dynemix significantly different from any other blockchain system. While in other cryptocurrencies all issued coins are distributed among the block producers, Dynemix distributes rewards among nodes that perform different functions, thus creating an absolutely new economic framework.

For a more detailed description of the role of the DOG reward system in the economy of Dynemix, please refer to the Economics chapter.

Rewards will be distributed by minters as the result of pseudorandom spot checks.

3) DOG as a solution to the reward distribution issue

A very important feature of DOG is that, unlike all other system support functions in Dynemix, DOG can be performed by even light clients (mobile devices). It serves the following purposes:

a) **Removing load from full nodes**

Full nodes perform various functions in the context of Dynemix/Liberdyne support. Unfortunately, due to the hardware and bandwidth limitations, we can delegate almost none of them to light nodes.

DOG, however, is a function that does not require powerful hardware or low latency. Considering that mobile devices will presumably form the majority of the Dynemix nodes, the load required by DOG will be distributed enough to significantly reduce the resource consumption of each dog. This also means that the load will not increase with the growth of the userbase, since the number of dogs will grow proportionally.

b) **Achieving greater reward distribution**

In the beginning of the white paper, we stated that one of our goals on the way to creation of a next-generation decentralized payment system was the following:

- **Newly issued coins should be distributed among as wide an audience as possible.**

The DOG reward system helps us achieve this goal. Since DOG is easily performed by any type of device, it fits the concept perfectly. If we make DOG rewards a large portion of the issued coins, this may allow the distribution of the issued dynes among almost all nodes of the network.

4. Secure anonymous tunneling across network (SATAN)

Direct P2P connections between users cannot provide full anonymity. Even if registration does not require any personal user data and all transferred messages are end2end encrypted, it is still possible to acquire the IP address of the sender and the recipient of any message and keep a record of user interactions, which in itself provides a lot of information.

This problem is also relevant for centralized messengers, since all connections are relayed through servers, which make it extremely easy for a person who has access to the servers to gather the information and use it for any purpose, including providing it to third parties.

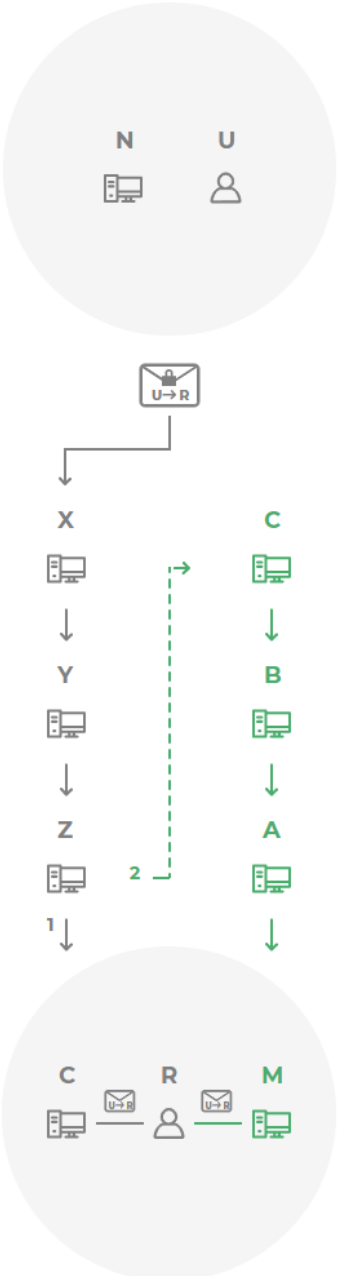
To ensure complete communication safety, we must apply a solution to hide the interaction between the users.

To accomplish this, we have implemented a secure anonymous tunneling across the network (SATAN) protocol that allows users to send and receive messages through a chain of relays, thus completely hiding their IP addresses behind the relay nodes. Furthermore, the relay nodes cannot know for sure which particular node is the sender or the recipient of the message.

The only fact that can be seen by a possible intruder is that the recipient's user account received some sort of message from an unknown account. Essentially, SATAN uses the principles of onion routing.

If the user wishes to hide all his activity, he may pay a fee to the system and turn on the SATAN protocol.

When node N run by user U is connected to the network, it chooses two chains of 3–6 relay nodes for incoming and outgoing messages.



1) Sending a message

a) When user U on node N wants to send a message to recipient account R, he gets information about which node is associated with user R from DHT. Say it is node C. U chooses three relay nodes for an output chain (X, Y, Z), sequentially encrypts the message, along with the metadata about the route, with the public keys of user R and of nodes C, Z, Y, X, and sends the encrypted data to node X.

b) X receives the message and decrypts it with its private key. It sees that the message contains some encrypted data and instructions to send the data to node Y, which it does.

c) Nodes Y and Z do the same as X.

d) C receives the message from Z and decrypts it with its private key. It learns that this is a message for account R. Since the owner of the node has the private key of account R, he decrypts the message and reads it.

None of the relaying nodes knows which particular nodes and which user accounts are the sender or the recipient of the message.

The owner of X may suppose that node N is the sender node; therefore, user U is the sender of the message, but it may also be just a relaying node, like X itself.

The owner of Z may suppose that C is the recipient, but it may also be just a relaying node, like Z itself.

Since nobody knows how many nodes U has actually chosen as relays, even if the adversary runs all nodes in the chain (X, Y, Z), he still cannot be sure that U is the sender of the message.

2) Receiving a message

If user R, along with user U, is using SATAN, then after exiting U's output chain, a message does not go directly to the node of R but enters his input chain instead.

In this case, R puts into DHT the information that he can be found not on the node on which he actually is (M) but on the first node of his input chain (C). Then, using his output chain, user R sends the following messages:

a) He tells node C that he can be found on B.

b) He tells node B that he can be found on A.

c) He tells node A that he can be found on M.

The message travels through his input chain and is finally delivered to node M, where R decrypts it with his private key.

Relay nodes from R's input chain know only that they are delivering some data to R, but they do not know the sender of the message or what node R is actually on. Node A may suppose that node M is the recipient node, but it may also be just a relaying node, like A itself.

Therefore, with the help of the SATAN protocol, any user can stay fully anonymous regardless of whether his or her interlocutor also uses the SATAN protocol.

3) Rewards for relays

Since SATAN is a paid feature, relay nodes will be rewarded for their effort.

Whenever a user wishes to enable SATAN, he or she sends an arbitrary number of dynes to the special system account. After minters add this transaction to a block, they update the user state in blockchain, setting a period during which the subscription will be activated.

The special system account accumulates all fees, and each block a portion of the accumulated dynes is distributed by the minters as the result of pseudorandom spot checks in a manner similar to DOG reward distribution, but unlike DOG, SATAN rewards are not coin-based and are withdrawn from the account mentioned.

5. Modes of operation: anonymous or social

Liberdyne was conceived as the ultimate means of ensuring the right to freedom of speech and of unlocking the potential of decentralized technologies, and that is where its name originates.

One of the key goals of the project is to provide the possibility of fully anonymous system usage and to secure personal data. The problem is that many features that consumers are used to are not compatible with such standards of security and privacy, so it is necessary to sacrifice one for the sake of the other.

At the same time, we understand that the majority of users do not care much about privacy and may be ready to trade it for better functionality. This means that we either have to compromise or offer two separate solutions – one for better privacy and one for a better user experience.

We have chosen to follow the second path and develop an application that can function as a tool for secure and anonymous communication or a social platform, depending on the user's needs.

A very important feature is that users of both modes will stay in the same single ecosystem and will be able to interact without any difficulties. Anonymous users can communicate with socialized users without any threat to their privacy.

As the features and preferences of these modes differ drastically, instead of just offering to let users tweak everything themselves, which can turn out to be a challenging task, Liberdyne will feature a simple switch that activates one of the modes.

When a user runs the app for the first time, he or she is offered the choice of either anonymous mode or social mode. After the user makes a choice, the app is automatically configured, and the interface is painted with the corresponding color to let the user know which mode is active at any given time.

1) Anonymous mode (blue)

A dark blue interface will indicate that the app is running in anonymous mode, which provides maximum security and privacy.

Once the user tries to access a feature that can threaten his or her anonymity, the app will show a warning. If the user proceeds with the feature, the color will switch to green, indicating that the user has quit anonymous mode. If the user wishes to return to anonymous mode, he or she can activate the switch from the preferences pane, which will instantly reconfigure the app.

Certain features, however, can deanonymize the user irreversibly. For example, if the user binds a phone number with the account, this information may be instantly collected by malicious actors, and removing the binding will not help him obtain anonymity again. In case such risky features are activated, Liberdyne will block switching to the anonymous mode to show the user that he or she cannot be completely safe further on.

Anonymous mode will feature the following specs:

a) **SATAN is always on.**

Considering that the SATAN protocol is a paid feature, anonymous mode will consume dynes, and users with zero balance will not be able to communicate with others in this mode. Transactions can still be issued free of charge, however (using the same chain of relays and keeping the user anonymous), so payment features will be available at any time.

b) **Features that require a streaming P2P connection are disabled.**

This applies to features that require data streaming, such as voice and video calls or any other features of this type that can be implemented in the future. Using relays to retranslate such types of data does not make sense, since the resulting delays may turn these features unusable.

c) **System support except relaying is disabled.**

System support functions require low latency, a good internet connection and stable transport routes. Providing system support while using a chain of random relays is technically almost impossible. In addition, it will put an unreasonably excessive load on relay nodes.

Performing relay functions in this mode, however, is not only acceptable but also enhances anonymity, as it creates additional noise in the user's traffic. Besides, turning the relay functions off makes the user more vulnerable to certain deanonymizing attacks, which is why performing relay functions in this mode is recommended.

d) **Binding a phone number to the account is disabled.**

Phone-number binding is a feature that poses the highest threat to anonymity. We actually do not like the idea of implementing this feature at all, but we understand that many users would feel uncomfortable without it, and rejecting this feature can greatly reduce the potential of Liberdyne's distribution.

2) Social mode (green)

A green interface will indicate that the app is running in social mode, which provides maximum functionality.

Social mode is designed to provide the level of user experience close to that of existing centralized solutions. Though it does not allow the full use of the potential of decentralized technologies, if we count on the mass adoption of Liberdyne and Dynemix, it will be necessary to offer familiar experience for the general audience.

Using Liberdyne in social mode does not provide the same level of privacy as in anonymous mode, but it may still seem sufficient for most users. All transferred data is end2end encrypted, and traffic obfuscation methods are applied in both modes, which means that going social does not make a user highly vulnerable to data analysis attempts, although metadata collection becomes less resource-demanding.

In addition, if the user performs relay functions, it creates a lot of traffic noise, making it extremely hard to analyze his or her connections. Finally, since Liberdyne employ standard Dynemix transport protocols, service data flows, which are hardly distinguishable from user-generated content transmissions, also increase privacy in the same manner.

6. System support tweaking

Liberdyne will be the default client for a Dynemix node and will include everything required to use the system and perform the full node functions. No special minting software or wallets will be needed (although third-party developers may come up with all kinds of specific software for different purposes).

Liberdyne versions for mobile devices (Android, iOS) will fully support only one service function – DOG – and may have limited support for relaying in the case of meeting technical requirements, while desktop versions will support the full capabilities of the system, including:

- Minting;
- Relaying;
- DOG;
- DCS.

This approach allows the improvement of the user experience and the creation of an interesting marketing concept.

According to the standard approach, which is used in most cryptocurrencies, the user should download a light client (a wallet) if he wishes to use the system only for payments, and a full node client if he wishes to participate in system support.

Liberdyne, on the other hand, is a complete solution in itself. The user does not need to think of choices – he or she just downloads the product and starts using it and everything will be set up automatically according to the user’s hardware capabilities.

At a first glance this does not seem to be much of a deal, but from the user's point of view the system starts to look totally different – it looks as if the messenger is paying users simply for using it.

The app will be configured so that users will not feel any significant inconvenience and at the same time, they will receive rewards for background system support. This may serve as a powerful marketing feature that will help us attract audience.

For a more detailed description of our marketing strategy and the role of rewards for system support in it, please refer to the Marketing chapter.

We will enable the support functions to be tweaked through Liberdyne's preferences. If the user wishes to share more resources to earn more rewards, he will be able to simply drag the slider and the app will be instantly reconfigured. It is especially convenient for mobile clients since the user will be able to precisely specify the amount of traffic and battery life that he is ready to share.

The user will be also able to prioritize certain functions or disable those in which he or she is not interested. If the user engages in minting, the app will automatically stop all other activities during the block-creation process to ensure its smooth operation.

7. Centralized services

Despite our goal to create a completely decentralized system, there are certain services that have a centralized nature and cannot be decentralized.

Since those services are not crucial for the system and do not affect the security and privacy of those who do not use them, we decided to implement them to improve the user experience.

1) Verified namespace

To verify accounts, a centralized trusted authority is needed, because verification requires establishing off-chain facts (like ownership of an account by a specific individual or legal entity).

In the early stages, this service will be administrated by the developers, but later we may consider transferring these authorities to a specially registered non-profit organization.

2) Phone number binding

Phone number verification requires the confirmation of ownership of the phone number by the account holder. Keeping the entire phone number base in the public domain is also not a good idea, which is why a centralized solution is required.

We do not like the idea of implementing this service since it can irreversibly deanonymize any user, but at the same time, we understand that many people are already used to such functionality and we must take into account the interests of the majority.

8. Decentralized cloud storage (DCS)

Certain functions of the messenger will require some sort of external storage of the user's data. Since we are making a decentralized system, we should not fully rely on third-party centralized solutions, and that is why a built-in decentralized storage protocol is required.

DCS will be able to provide users with confidence that their sensitive data is stored persistently and that access to the data cannot be arbitrarily restricted.

Liberdyne requires external storage for the following types of data:

a) **Contact lists**

Centralized messengers either store user contact lists on servers (Telegram) or use contacts stored on the user device (WhatsApp and others), which in turn are backed up in the clouds of OS developers and can be restored from user's Google/Apple account.

Keeping such a type of information in blockchain is not a good idea, since it will greatly increase the weight of the account states, which is why we can either offer users the ability to employ third-party centralized clouds or use the DCS solution.

b) **Generated content backup**

Centralized messengers either store user-generated content (messages, files, pictures etc.) on servers (Telegram) or use third-party cloud storage (WhatsApp and others).

Since dialogs may also be important to users, we should provide secure storage, which can also be achieved by keeping backups in DCS or third-party clouds.

The Dynemix DCS protocol is still in development. More details about the protocol will be provided when ready.

VII. Economics of Dynemix

The economics section is being revised and will be available later as a separate paper.

VIII. Decentralization

1. Understanding decentralization

Decentralization is the main idea behind cryptocurrency and the very reason for blockchain technology's existence. Despite this fact, there is no established definition of the term, and the very notion of decentralization in terms of the cryptocurrency industry is still a matter of lively debate, with no clearly defined approaches having been developed.

All this has led to heavy abuse of the term by developers, who started to claim they had reached decentralization only due to the fact that their systems were maintained by more than one node or due to the mere replication of databases.

Since we mentioned decentralization as one of the fundamental properties we seek to achieve, we should cover our comprehension of the subject and explain what exactly we are striving for.

We are going to take a more practical approach and will not conduct research to describe decentralization as a whole. Instead, we will concentrate on the meaning of decentralization with regard to blockchain technology.

We are aware that we are discussing a controversial matter and we do not claim to be establishing the absolute truth but rather to share our own view, whether it is supported by the community or not.

The distinctive feature of our approach is that it is based on social factors more than on technical ones. We have observed many studies that have tried to understand decentralization exclusively with the help of a computer science framework, but we believe that decentralization is a social phenomenon that cannot be efficiently defined only with technical terms.

Though the idea of developing a sort of a decentralization equation, which will allow us to input a set of variables and calculate the precise level of decentralization, looks tempting, we do not find it possible for the reasons stated.

Let us get to the point and try to define what exactly a decentralized blockchain system is.

A decentralized blockchain system is a blockchain system designed to deprive a single individual or a coordinated group of individuals of the opportunity to affect the system's operation at their discretion.

> It is worth mentioning that, though we are speaking about blockchain, most of the further statements can also be applied to DAG-based cryptocurrency platforms and any other possible types of DLT systems.

This definition is vague, but this is a consequence of the fact that the border between centralized and decentralized is very conditional. Let us investigate the matter in detail and try to clarify the definition as well as develop an approach that could allow a practical assessment of the decentralization level.

1) "Individual"

Problems begin to emerge directly from the basic terms to which we refer in our definition. Any blockchain ID (account, public key etc.) can be controlled, either by a person who acts on his or her own behalf or by a person who acts on behalf of some incorporation. From the point of view of the entire system, it is impossible to distinguish who actually is hiding behind any particular ID.

On the other hand, a single individual can create multiple IDs within one blockchain, and again, the protocol rules cannot be designed to reliably detect such behavior.

That is why when we talk about a blockchain user or a node, we should keep in mind that it can represent an incalculable multitude of individuals or on the contrary be just one of many representations of a single individual. It is impossible to settle this issue in a permissionless pseudonymous distributed system, so we can only accept it a fact.

For these reasons, power in DLT systems is not distributed evenly among a set of entities but derived from the resource assigned to each entity. Despite the fact that voting is carried out according to the amount of resources, which means that a single entity may gain more voting power than an arbitrarily large group of opposing entities, this concept is designed to prevent the concentration of voting power in the hands of a single individual or a coordinated group of individuals by being based on the assumption that the required number of resources is high enough to be unlikely to be concentrated in one set of hands.

2) "Opportunity to affect"

Ideal decentralization implies the impossibility of decision-making by a coordinated group. The problem is that there is always a group (larger or smaller) that makes decisions, and, theoretically, any group can start acting in a coordinated manner. This means that no blockchain system can be absolutely decentralized and we cannot apply a simple binary approach. Each system can only be placed somewhere on the scale between more decentralized and more centralized.

This is also the point that makes the definition indistinct. It is hard to define a particular set of conditions under which such an opportunity emerges, especially given the variety of current blockchain designs and the unclear possible vectors of future development.

We can conditionally outline two main factors that define the presence of the opportunity:

- **Quantitative.** A number of entities or resources is required to produce an effect on the system. The system that can be influenced by 10 actors is obviously more centralized than the system that can be influenced by 1000 actors; it is hard, however, to define the exact threshold.

- **Qualitative.** An image of an average actor. When Bitcoin blockchain was launched, Satoshi expected the network to be supported by common users who perform PoW computations with CPUs, thus providing the required protection. During its infancy, an image of an average Bitcoin miner generally matched expectations. With the appearance of ASICs and the growth of popularity, however, mining turned into a purely professional commercial activity, significantly transforming the portrayal of an average miner.

3) "System operation"

This term consists of several elements of the system and a set of properties that define these elements. Considering current blockchain implementations, we can propose the following classification:

- **A state transition.** A transition from state A to state B by agreeing on an input value (a set of transactions) and running a predefined function that generates an output (commonly in the form of a block) accepted by the participants of the network. The affected properties of state transition are safety (can be decomposed into consistency and validity) and liveness; censorship resilience (or fairness) should also be considered.

- **Data storage.** The main property is availability, which is secured by the redundant replication of the database.

- **Development.** How the protocol is administrated and updated. From the point of view of decentralization, the degree of influence on decisions that affect the rules of the protocol should be assessed.

We would say that, speaking of the overall decentralization level of a blockchain system, decentralization of the state transition process seems to be the fundamental concept-defining matter, while decentralized storage is an auxiliary matter.

With regard to decentralized development, we should state that this is the point where things become really complicated. While conceptually it seems that we should tend to decentralize the development process as much as possible to fully achieve the overall decentralization, in practice it is not that simple.

4) "Designed to"

There can be two approaches to the continuity of the decentralization level of a given system.

- According to the first approach, the decentralization level is a fluid property that may vary depending on how the set of the system's properties changes over time.

  If we apply such an approach to Bitcoin, we may conclude that it was more decentralized at the start and became more centralized after the emergence of mining pools.

- We support another approach, according to which the decentralization level is constant and defined by the architecture (unless it changes) and the initial state, but not the actual state at any given time. What is important is not how much the system is decentralized at the moment, but how centralized it can theoretically become due to the protocol rules or other factors.

  Returning to the example of Bitcoin, we may state that according to our approach its decentralization level on the start was the same as now, despite the fact that mining pools did not exist at the time, but considering that their emergence could be predicted.

Following the first approach can quickly become a situation of walking on thin ice if we treat the system as a decentralized one, while it can predictably turn completely centralized in the blink of an eye.

But the second approach is not flawless, either. There can be certain barely predictable conditions that can appear unexpectedly and change the attitude to the decentralization level.

In addition, in most cases our approach cannot be directly applied to the infancy stage, as blockchains are generally initiated by developers, which often have all the power concentrated in their hands for a short time, until at some point the power is distributed among the participants. Such an approach does not necessarily indicate centralization, as long as the power transition procedure does not last unreasonably long.

2. The difference between decentralized and distributed

We came across several approaches that refer to decentralization as the distribution of a blockchain (as a database). This may come from the fact that blockchain systems are basically defined as distributed state machines. The problem is that the terms “decentralized” and “distributed” are used in different contexts and sometimes even interchangeably.

We consider decentralized to be a degree of distributed.

We suppose that any distributed system becomes decentralized only after it reaches a certain state of distribution that allows the achievement of the properties specified in the definition. We may state that all decentralized systems are distributed, but not all distributed systems are decentralized.

Let us explain what we mean in the example of the above-mentioned database distribution. The general purpose of database distribution itself is to provide fault tolerance, high availability and a decrease in access latency. In relation to blockchain, the first two features seem to be the priority ones. Indeed, being distributed, a blockchain becomes resistant to faults, data loss and unauthorized censorship attempts. These are useful features, but are they the main idea behind the blockchain concept?

Presumably, Satoshi created Bitcoin in opposition to traditional payment systems, which also feature database and processing distribution. So, why do we call VisaNet centralized and Bitcoin decentralized?

The main issue of conventional payment infrastructure is that all vital actions, including currency emission, transaction approval, compulsory withdrawal of funds, provision of access to the system etc. are within the competence of appointed entities. These entities are often suspected of abuse of their powers in the pursuit of the interests of minority elite groups. Distributed technologies that are used in conventional payment systems do not help solve these issues, but this is due to the fact that they were not originally meant to solve these issues – they were implemented for different purposes.

The Nakamoto consensus protocol was invented in an attempt to create a payment system in which no entity or group was capable of making vital decisions, and this was the nervus rerum. This means that, with regard to blockchain technology, distribution starts to serve a specific primary goal, which is why we need a special approach to our understanding of decentralization.

For this reason, we believe that decentralization cannot be defined through the mere distribution of the database or another single component. We need a certain set of distributed components, which are distributed to a certain degree, to ensure the particular behavior of participants and achieve the stated goals to call a system truly decentralized.

3. Decentralized state change

Blockchain systems are accessed via special entities that represent users or user groups. Commonly, a cryptographic key pair is used for this purpose, whereas a public key is a basic entity that identifies a user.

Such an entity being an abstraction does not allow the quantitative factor of the desired level of decentralization with regard to state transition to be determined: in other words, we cannot measure the decentralization level of a blockchain system based on the number of entities needed

to make decisions, because, for the reasons mentioned above, “number of entities” has no practical meaning.

At the same time, we cannot simply substitute the abstract term “entity” for the more practical term “individual,” since it is impossible to collate the number of entities against the number of individuals behind such entities in a permissionless, pseudonymous blockchain system. That is why we need to add an additional property that can allow quantitative assessment.

To accomplish this task, it was proposed to use the number of resources in possession as a quantitative factor of the decision-making process.

The main idea behind this approach was to try to ensure that no single individual or coordinated group of individuals could obtain enough resources to make decisions on their own. The more resources are needed to change the system state and the harder it is to obtain this number of resources, the more decentralized the system theoretically is.

To date, we have two types of bounded resources that are commonly used for this purpose:

- **Computing power.** The resource that is counted in PoW protocols. The level of decentralization can be measured according to the hashrate of the hardware.
- **Units of the system.** The resource that is counted in PoS protocols. The level of decentralization can be measured according to the number of coins in possession.

There have been several other proposals of resources to count, but none of those concepts has received any significant support to date.

There are also concepts involving reputational factors, but they are mostly used as an auxiliary coefficient applied to the number of resources in the possession of the user. Although it is technically possible to build a system based solely on reputational voting factor, the safety of such a system is highly questionable, and the concept requires additional research. To a certain extent, delegated PoS may be also considered a reputation-based design – although voting for block producers employs resources, the only thing that is actually at stake is reputation. For now, a resource-based approach seems to be the only sufficiently Sybil-proof design.

The safety guarantees of blockchain protocols are based on the assumption of the majority (supermajority) of voting power controlled by the nodes that behave properly.

Liveness guarantees are not necessarily based on the number of resources in the possession of honest nodes, as they are relevant only to BFT-based design, but it should be also taken into consideration.

To assess the decentralization level, the following parameters should be clarified:

a) **The number of resources required to influence a state transition.**

   Both PoW and PoS designs assume certain thresholds of voting power that allow the state transition process to be influenced (by violating safety, liveness or fairness or by affecting the system another way). Generally, we may state that, due to the fundamental upper and lower boundaries, regardless of the particular consensus design ,in various ways the system operation can be affected by actors who control from $\frac{1}{3}$ to $\frac{2}{3}$ of the voting power (although there are few exceptions).

   There can be, however, certain scenarios of possible selfish behavior that involve a lesser amount of the resources (e.g. selfish mining in PoW-based protocols, forcing a transition from the fast to the slow execution path in hybrid PoS protocols etc.), which should also be taken into consideration.

   Although the relative share of resources required for decision-making is more or less equal in most designs, the proportion of the total resource-backed pool of decision influencers in relation to the total value of the system may vary significantly. The total value of the system’s

circulating supply correlates with the possible expected return from the selfish behavior of rational players, which is why the notion of a consensus majority or supermajority may have a completely different meaning in practice. This issue should also be considered.

b) **How resources are acquired.**

Safety and liveness depend on the chance of one actor's or a closely related group of actors' obtaining the required amount of the resources. This includes both the protocol rules and the method of initial token distribution (for PoS systems). The protocol rules define how much of the resources (in computing power or stakes) is needed to make a decision and how such resources can be gained by participants. The initial token distribution procedure determines the degree of token distribution or, in other words, the possibility of the minority's controlling the majority of stakes right from the system's start.

The matter of token distribution is as important as the protocol rules. There are numerous so-called "decentralized platforms" that have their tokens distributed via private sale among a small number of participants. Other developers leave themselves a huge share that allows them to manipulate consensus. Some developers even do both. Although technically such platforms may seem to provide decentralization at the protocol level, in practice all decisions can be made by a small predefined group of initial token holders, which makes such systems completely centralized.

c) **How low the entry threshold is for block producers.**

The more individuals involved in the process, the harder it is to make such a process organized. With regard to blockchain systems, this means that the more people involved in the block production process, the less likely they are to unite in a group and start acting in a coordinated manner in pursuit of common interests that are contrary to the common interests of the entire community.

To achieve these conditions, it is crucial to lower the hardware and bandwidth requirements as much as possible. The perfect result is the opportunity to run a state transition on desktop-class hardware with average bandwidth, even with a significant transaction flow (assuming a PoS design, of course).

Higher requirements lead to the professionalization of block production, which not only reduces the number of participants, but also affects the quality – when block production turns into a professional activity, producers, who begin to pursue purely commercial purposes, are more likely to form cartels. On the other hand, when the process is executed by common people and no significant investment is needed, altruistic behavior can prevail and coordinated adversarial strategies become harder to execute. Though we cannot rely exclusively on altruistic behavior and the system should also be designed to be safe in the case of the predominance of rational players, increasing the number of possible altruistic players can greatly benefit the system's decentralization beyond the assumptions of game theory.

This is why keeping the block production process available to common people is a critical task on the way to creating a decentralized blockchain system.

There can also be other obstacles to participation in the block production process apart from hardware and bandwidth requirements. The most common example is the stake size in PoS systems. If we set the minimal stake to a level equal to one million USD, this will assuredly exclude common people from minting regardless of the hardware requirements.

4. Decentralized data storage

Decentralization of the database is achieved through redundant replication and geographical distribution.

We do not suppose that the level of database distribution should be considered a measurement of the level of overall decentralization of the system, because it always follows the distribution of the

state transition process. Block-producing nodes must always store information, allowing at least the establishment of the current system state. Otherwise, replicated system state transition is unachievable.

Given this fact, we can proclaim that crucial data distribution is achieved together with decentralization of the state transition since the current state of the system is required to participate in the procedure. Storing such a state on external nodes makes no sense from the point of view of time and traffic optimization; therefore, we have never come across a blockchain system design that separates state transition from state storage. We should note, however, that a sophisticated scheme can be envisioned in which the collation of the state transition output is separated from the state storage, and we cannot reliably claim that our statement is fundamentally irrefutable.

At the same time, this does not work the other way around. Data storage distribution alone does not necessarily lead to the decentralization of the state changing procedure. Otherwise, we could conclude that MasterCard's Banknet is a decentralized platform since its transactions database is replicated. This means that the level of database distribution does not help determine the level of decentralization of the system.

In PoW-based systems, however, overall data availability is not bounded to state transition, as the nodes that keep the record of the blockchain do not necessarily engage in mining. For this reason, we may asses the following additional factor:

a) **The hardware and bandwidth requirements to maintain the database.**

   The less onerous the requirements are, the more instances of the database that can be expected to be available simultaneously. This question strongly correlates with the question from the previous section concerning the requirements to run a block-producing node, and, in most cases, investigating the latter is sufficient.

5. Decentralized development

There can be two different approaches to the method of platform development after release. Let us explore each using the example of the two most popular blockchain platforms to date.

1) Development by community

This is the method undertaken by Bitcoin. Shortly after Bitcoin's release, its creator, Satoshi Nakamoto, disappeared, and since then the platform's development (at least its main fork) has been carried out by the Bitcoin Core team – a group of independent developers who did not participate in the Bitcoin protocol or the software's initial creation.

What advantages and disadvantages of this approach have been revealed?

a) **More decentralization is expected**

   Since, theoretically, there is no central authority, which can have a huge effect on consensus, such an approach looks more decentralized, and in this way, it better fits the ideas behind cryptocurrencies.

b) **The decision-making process is complicated**

   At the same time, the absence of any central authority creates a problem – it is much harder to adopt decisions seriously affecting the platform's architecture. From the social point of view, a community usually acts inertly and passively. If we rely on a voting process, it may take a lot of time to approve any decision.

c) **High probability of a fork appearance**

Another problem is that, without a central authority, all decisions will be criticized, and if the division of opinion is high enough, then it will lead to the creation of a fork. We already have Bitcoin (the parent fork), Bitcoin Cash (which has also split into two independent forks), Bitcoin Gold, Bitcoin Diamond, Bitcoin Private and Bitcoin Interest, as well as lesser forks. On the other hand, Ethereum, which is still supported by its creators, has only one notable side fork – Ethereum Classic.

d) **The development process eventually becomes centralized**

Although, theoretically, such an approach should provide more decentralization, in practice the development process cannot be carried out in a decentralized manner. Even if we managed to involve a significant number of contributors, there must be a leader who will put it all together. Therefore, in an attempt to avoid chaos, we will inevitably end up with a central authority.

Another obstacle to reaching the decentralization of the development process is the requirement that the participants have high competence, which results in a high entry threshold.

To become a miner of Bitcoin, one will theoretically need at least one simple ASIC, an internet connection and a power socket – things that many people can afford. Considering that configuring mining software is a feasible task even for a person with basic knowledge, we can conclude that prominent mining decentralization is achievable (we are not taking into account the cost optimization issues here, though).

An average Joe cannot become a blockchain developer, however. It takes a lot of knowledge and experience in the innovative technology to be able to join the development process, and this is something that only a few possess.

These factors lead to the conclusion that decentralized development is nothing more than a utopia.

Indeed, if we imagine that the development process is carried out by thousands of people, we will see no way of making it organized and effective without an appointed leader or group of leaders. Otherwise, the development process will turn into a tug-of-war, which will end up with a few winners occupying the podium anyway.

Due to human nature, such leaders may eventually start pursuing their own interests, which will lead to a fairly predictable outcome.

If we look at the current state of Bitcoin, we can see that, though the problem of increasing block size has been being widely discussed for many years, the Bitcoin Core developers have maintained the size as it is, as determined by Satoshi before quitting. This has raised a lot of negative judgment toward the developers. Some users accuse them of pursuing their own interests while ignoring the community's interests. Such judgments are motivated by the alleged developer's strong connections with major mining pools and mining hardware manufacturers – since a smaller block size creates competition among the transaction issuers, which increases transaction fees, it is more beneficial for miners to keep the block size intact.

The developers insisted on implementing SegWit instead of a larger block size, which helped a bit, but did not change the situation dramatically.

If Satoshi was still around, would the situation be the same? Supposedly, he considered the increase in the block size's a method to scale the system, and most likely he would have supported it. Of course, this does not mean that Satoshi's presence would help the community win the war against miners, had they opposed the decision to increase the block size, but at least it could have given the community a chance.

Meanwhile, the “decentralized” development has led to a situation in which the interests of mining pools are put above the interests of the entire community.

Another fascinating event in the development of Bitcoin took place in September 2018, when several developers received information about a serious bug that could allow a DOS attack, and – what is more important – a possible attacker could issue new bitcoins by claiming the double-spent value.

The developers issued a patch immediately, but what is most interesting is that only the information about the DOS attack vulnerability was disclosed, not information about the vulnerability to inflation (for security purposes). The latter was disclosed only after most miners had moved to the new version and the threat had been eliminated. Though we should not criticize the developers for behaving this way, since they obviously acted for the sake of the whole community, this situation shows us that a small coordinated group of developers can easily force the community to instantly accept any undocumented features, which does not coincide with the logic of decentralization.

This leads to the conclusion that following the path of development by the community eventually makes the process centralized regardless of the initial intentions.

2) Development by creator

In the method undertaken by Ethereum, Vitalik Buterin and his team have been developing the platform right from the start.

a) **Faster platform development**

As we can see now, the presence of an authority can lead the community in a certain direction, which has a positive effect on the pace of the platform’s development. Ethereum has gone through many major technological changes, including modification of the consensus protocol. There is even a non-zero probability that one day we will finally see Ethereum 2.0 released.

b) **Unpopular decisions may be lobbied for**

The other side of the coin is the possibility of forcing unpopular decisions, and such decisions not being rejected by the community due to the developer’s strong authority, which can outbalance even a majority disapproval.

c) **Fork prevention**

On the other hand, the possibility of forcing unpopular decisions prevents a platform from forking. Even if those who disagree with the developer’s course start a fork, most of the community will be highly likely to support the main fork unless the developer messes up really bad.

Speaking of Ethereum, the most vivid illustration of these statements is the DAO fork. Events following the DAO hack have allowed us to learn two lessons from this situation – on the one hand, the creator can have enough influence to force a decision that goes against the rules of the system, while on the other hand, the decision made by the creator, even if heavily criticized, can set the sole direction in times of uncertainty, preventing the system from shattering into lesser forks and falling into oblivion.

3) Conclusion

All this leads to the conclusion that decentralization of the development process is not a matter of particularly high importance. Distributed development can cause more troubles than it solves, so it is highly uncertain whether we should interpret it positively, especially considering that in practice it is almost impossible to truly decentralize the development process. On the other hand, centralized development is mostly beneficial to the platform, and after all it may become even more acceptable.

We do not support a “decentralize everything” approach, since decentralization is not the ultimate goal, but merely the means of reaching other goals. If it becomes to the detriment of the system, there is no need to stick to it no matter what.

There is one important question, however, that we should investigate to evaluate decentralization with respect to the development process:

a) **Obstacles to creating a fork**

With regard to the development process, what is important is not the actual distribution of the process itself at any given time but the possibility of the free development of an alternative software, which includes the fork creation.

This means that the software should be free and open source. This will provide an opportunity to develop alternative forks or clients in the case that community does not support the current developer’s course, which will actually ensure the plurality of opinions and the desired decentralization.

If the developer protects the software by copyright, patent or any other available methods he or she will monopolize the opportunity to change the rules, and the community will have no option but to accept the developer’s choices, which means that such a system cannot be considered decentralized.

6. Overall decentralization level assessment

Having formulated all the questions needed to assess the level of decentralization of each component of the system, we can now develop an approach to assessing the overall level of decentralization.

We can follow one of two main approaches to solving this task:

a) **Calculate the median.**

We can take an assessed level of each component and discover an average value for it, which will show the overall level of system decentralization. For example, if the system is more centralized in some respects, but provides more decentralization in others, these aspects will balance each other and provide a medium overall level.

b) **Determine the bottleneck.**

The overall level can also be estimated through its weakest point. We find this approach more reasonable in practice.

An important property of decentralization is that any component can become a point of failure. There are some existing projects that pretend to be decentralized by having certain decentralized components, but at the same time, they restrict creating forks or empower a single entity with censorship capabilities. Despite that they actually provide some decentralization from certain points of view, they obviously cannot be called decentralized overall.

This is why we believe that each component must bear a sufficient level of decentralization for the system to be considered decentralized.

7. Decentralization of some popular blockchains

Let us pick up three popular blockchain platforms and evaluate the decentralization level of each by applying the approach described above.

At the time of the writing of the white paper, Bitcoin and Ethereum were the undisputable industry leaders, which is why they should definitely appear on our list. The choice of a third competitor is not so trivial, however.

From the point of view of market cap, the next competitor is Ripple, which is obviously centralized and is not positioned as a cryptocurrency initially. Bitcoin Cash and Litecoin are too close to Bitcoin in design, which makes their analysis unreasonable. For these reasons, EOS, being a recent project with a significantly modified design, seems to be the most relevant choice.

1) Bitcoin

Bitcoin is the industry leader and the first platform that can be described as a decentralized payment system. It would be logical to use it as a benchmark for any other systems and assess whether they managed to provide better decentralization.

State transition

The Nakamoto consensus does not feature any particular liveness threshold, and as long as the system holds synchrony, the main properties that may be influenced by a coordinated group are safety and fairness.

We have already evaluated Bitcoin's safety threshold and concluded that it would take about 0.3% of Bitcoin's total market value to violate safety. It should also be taken into account that these expenses are not irrecoverable.

Considering the importance and value of Bitcoin in terms of the whole blockchain industry, we may take this numbers as a default benchmark and presume that it is equal to a medium level of decentralization.

As for fairness, Bitcoin fails to provide an acceptable level of decentralization.

It is widely known that Bitcoin mining is mostly controlled by several major mining pools. As of the beginning of 2019, five pools controlled >50% of the total hashrate. This means that the coordinated actions of five actors could result in any kind of selfish behavior.

Bitcoin protocol is vulnerable to selfish mining and censorship. Spy mining does not seem likely to occur, since Bitcoin has quite a long block time and small block size, due to which the spy mining strategy does not bring any tangible benefit.

Censorship may become a problem, however. Since only a few actors produce the overwhelming majority of blocks, they may come to an agreement about the set of transactions to include in the blocks.

Assume that there are two competing services that use Bitcoin as their primary (or the only) payment method. One bribes all the major mining pools to reject all the transactions of the other one. The second service's transactions start being processed extremely slowly, and eventually the service loses the competition for this reason. This is just one of many possible scenarios involving censorship abilities.

Hardware and bandwidth requirements (entry threshold)

All PoW-based systems in the mature state require expensive professional hardware and a very meticulous approach to economic efficiency to keep mining activity profitable. We can confidently assert that mining nowadays is a purely professional enterprise wherein altruistic behavior from participants cannot be expected, and we should conclude that decentralization from this perspective is below average.

Though the bandwidth requirements are low, this does not significantly help the situation.

As for database storage, since Bitcoin does not feature sharding and uses the UTXO transaction model, it requires the full blockchain to be stored on each node, which leads to the high storage space requirements. With the current block size, the bandwidth and computing power

requirements are low, so it is easy to run a full node of Bitcoin even at home if the user is ready to sacrifice disk space.

If the system load grows significantly, however, database distribution may drastically decrease, although that would require a hard fork to extend the block size limit.

From this point of view, Bitcoin does not provide a sufficient level of decentralization, assuming that the system is scaled commensurately with the growth of demand.

Development

Bitcoin already features many forks, and their number is constantly growing, so there are no obstacles to forking. This may lead to the conclusion that from this perspective Bitcoin is highly decentralized.

Bitcoin's architecture (as well as that of any other PoW system) has one significant peculiarity, however – mining power can be freely applied to any system that uses the same hashing algorithm, which obviously includes all the forks of any given blockchain.

In the beginning of 2019, the Bitcoin hashrate reached approximately 40 EH/s, while the hashrate of Bitcoin Cash (the second largest fork of Bitcoin) reached only about 1.5 EH/s. Such a huge difference shows that in the case that major miners consider a fork as threatening their interests, they can instantly use just a small fraction of their hardware to execute double-spend attacks on any undesired competing fork, thus depriving it of the users' trust and of the possibility of further existence.

This example shows that the possibility of fork development depends completely on the attitude of a small group of mining pools or even single powerful miners, and therefore the level of decentralization is low.

Overall level

We can conclude that, due to the inherent shortcomings of PoW-based consensus, which have led to the emergence of mining pools and concentration of power in the hands of a small group of actors, Bitcoin's decentralization is below average.

2) Ethereum

Ethereum represents a different concept of the use of blockchain technology – a smart contract platform or a decentralized Turing machine. It also features an accounting model instead of Bitcoin's UTXO approach. Considering that Ethereum appeared much later than Bitcoin and has many technical differences with it, it is interesting to learn whether it has made any significant step toward providing a higher level of decentralization.

State transition

The consensus protocol of Ethereum is derived from the Nakamoto consensus, so it holds roughly the same properties.

Ethereum's safety guarantees are based on the "honest majority" assumption. As we have already figured out, honest majority is a highly conditional notion, and it is better to evaluate the practical ratio of the majority of voting power and the market value of the system. Let us apply the same approach as we did for Bitcoin to make the comparison fair.

The lowest registered hashrate occurred in the Ethereum network on Feb 3, 2019. The hashrate reached 139.15 TH/s on that day, which means that

$$139.15 \div 2 = 69.75 \text{ TH/s}$$

was enough to perform a double spend. The total market cap reached about 11.5 billion USD. It is much harder here than in the case of Bitcoin to estimate the price of the hardware that could produce the required hashrate, since there can be many possible configurations, but let's pick up a

common mining rig that is powered by 8 AMD RX580 GPUs and costs roughly $2,100. Such a rig produces approximately 240 MH/s, so we would need about

$$69{,}750{,}000 \div 240 = 290{,}625$$

mining rigs. The overall expenses for an attack would reach about

$$290{,}625\ x\ 2{,}100 = 610{,}312{,}500 \text{ USD}$$

which gives us the Ethereum safety coefficient of

$$610{,}312{,}500 \div 11{,}500{,}000{,}000 = 0.053$$

The result is actually impressive in comparison to Bitcoin. Such a difference can be explained by the significantly higher emission rate of ether, which makes mining more profitable and leads to an increase in the hashrate, as well as in several other factors.

As in the case of Bitcoin, the expenses are partially recoverable. Moreover, whereas Bitcoin ASICs actually have near-zero applicability in any task except Bitcoin mining (or that of any other SHA-256 based platforms), the GPUs used for Ethereum mining are standard PC hardware, and it should be much easier to resell them. We should note, however, that with regard to our example it would be a non-trivial task to sell a couple million RX580 graphic cards on the secondary market.

Overall, it is clear that, with the current emission rate, Ethereum provides higher safety guarantees than Bitcoin, which leads to the conclusion that Ethereum's decentralization level from this perspective is higher. The relative emission rate is constantly decreasing, however, and at some point, the double-spend resistance of Ethereum is likely to lower to the level of Bitcoin or even further, although we cannot be sure that this will ever happen, since the scheduled protocol modification should change the situation completely.

As for fairness and possible selfish behavior, Ethereum has all the same problems as Bitcoin. In addition, Ethereum is vulnerable to spy mining, which could actually be witnessed in the system in 2018. We can state that the level of decentralization from this point of view is low.

Hardware and bandwidth requirements (entry threshold)

As with Bitcoin, Ethereum requires a professional approach and expensive hardware to participate in mining. Most common mining rigs include 6–12 GPUs, which makes using a standard PC for mining economically unfeasible, so we should conclude that decentralization from this perspective is below average.

As for data storage, there are two types of database-storing nodes – a full node and an archive node. Requirements for the latter are much higher, but archive nodes do not provide additional resilience for the system and are mostly needed for statistical and research purposes. To assess decentralization, we should count only full nodes, which are crucial for the system's operation.

Since full nodes need to process smart contracts in order to verify blocks, their hardware requirements are rather high. This is a natural characteristic feature of the Ethereum concept – as a decentralized computer, Ethereum obviously requires computing.

Ethereum uses the accounting model instead of Bitcoin's UTXO model, so it is better optimized in terms of storage. On the other hand, Ethereum architecture requires all smart contracts and their states to be stored on a node, which leads to an increase in the occupied disk space. As a result, Ethereum's storage requirements are comparable to those of Bitcoin.

Overall, we can assume that in the case that the system is scaled commensurately with the growth of demand, in time hardware requirements will become rather high, which will reduce the number of full nodes and database instances unless some economic incentives are adopted. The level of decentralization under a heavy load will be below average.

Development

The situation is exactly the same as Bitcoin's, so we can state that the level of decentralization from this perspective is low.

Overall level

We can see that, although in some respects Ethereum has managed to secure a higher level of decentralization than Bitcoin, its overall level is still below average, for many problems remain unresolved. We should keep in mind, however, that with the introduction of Ethereum 2.0, all of the properties we have analyzed are expected to change.

3) EOS

EOS is an example of a more recent project. It uses the DPoS consensus protocol and is basically a smart contract platform like Ethereum, with an emphasis on fast transactions, high scalability and governance. Let us see whether the developers have managed to provide sufficient decentralization along with the mentioned properties.

State transition

EoS architecture is radically different from most blockchains due to the fact that not only resources defy power within the system, but also social and political factors, which makes our standard approach not fully applicable here.

In EOS, consensus is reached by the votes of 21 block producers. Due to the majority consensus threshold and the longest chain rule, the coordinated actions of only 11 pre-known actors can result in any type of harmful selfish behavior, including censoring and double-spending. Moreover, due to the essence of the DPoS algorithm and its political nature, no resources in possession are actually required to become a block producer and the only thing a possible attacker risks losing is his or her reputation. This means that we cannot calculate a safety coefficient as we did in previous cases.

In addition, EOS is conceived as a governed system and features an entity called the EOS Core Arbitration Forum (ECAF), which is endowed with extremely broad powers, including the ability to reverse confirmed transactions and suspend user accounts.

Although the system was launched not long ago, there have been several reports of the practical use of these powers.

Since these opportunities can be used not only for the sake of the entire community but also for the benefit of any entity or group, this leads to the conclusion that EOS completely fails to deliver decentralization from the point of view of resistance to selfish behavior.

Overall level

Since we already see that EOS's state transition process is centralized, there is no sense in researching other features, and we can conclude that the overall decentralization level of EOS is extremely low, so it cannot be considered to be in the same league as decentralized platforms.

8. Decentralization of Dynemix

During the development of Dynemix, we paid a lot of attention to the matter of decentralization.

As we have already figured out, first-generation blockchain systems based on PoW protocols turned out to be not as decentralized as expected. At the same time, more recent projects' developers tend to trade decentralization for speed and scalability.

We do not support such an approach, and we have put a lot of effort not just into ensuring a sufficient level of decentralization but also into making Dynemix one of the most decentralized blockchain systems to date.

Let us explore what has been achieved with the help of our approach.

1) State transition

a) Safety and liveness thresholds

Since Dynemix features sharding and a two-layer consensus, the question of safety and liveness thresholds is complicated.

The system makes progress as long as at least one shard-block at each height gets approved and $\frac{3}{5}$ of minters are honest and online. We may intuitively assume that the share of the global stake pool that is required to stall the protocol lies between $\frac{1}{3}$ and $\frac{2}{5}$. At the same time, a consistent violation of safety would require roughly $\frac{2}{3} - \varepsilon$. A more precise assessment can be provided after the test period.

All in all, we may state that, from the point of view of raw numbers, the safety and liveness thresholds are close to most asynchronous BFT protocols.

We have applied a set of measures, however, that ensure the growth of the global stake pool size, including

- setting a low minimal stake boundary;
- securing low hardware and bandwidth requirements;
- allowing funds to be unbonded quickly;
- setting a high issuance rate;
- adopting a model of weak responsibility.

With the adoption of these measures, the share of resources needed to affect the state transition in relation to the market cap of the platform is expected to be significantly higher than in other blockchain systems.

b) Participants

Resisting professionalization of the state transition processing was one of the key goals of our project. Although we needed to provide sufficient speed and scalability, at the same time we managed to set a low participation threshold.

The involvement of ordinary users is ensured by providing the opportunity to participate at will without the need to hold uninterrupted availability (the model of weak responsibility) and having a low minimal stake threshold. Furthermore, the entire economic model of Dynemix is designed to favor regular users instead of minor elite groups.

c) Initial token distribution

Given our devotion to decentralization, we do not plan to distribute the initial token volume in a manner that can allow the concentration of a large share among a small group. The bulk of tokens will be distributed via a prolonged public sale at a market price, which seems the optimal way to assure the widest distribution.

d) Issuance model

Even if certain entities manage to acquire a large share of tokens at the initial stage, it will not help them keep control of the system later on. Unlike most PoS designs, which make the rich get richer (because the largest stakeholders obtain most of the issued coins), Dynemix features a unique issuance model that disperses the vast majority of the issued dynes among a large number of participants (helicopter money), thus constantly reducing the relative size of the initially acquired shares and preventing the concentration of power.

2) Hardware and bandwidth requirements (entry threshold)

During the development process, we devoted a lot of attention to optimization. Since we intended to create a truly decentralized system that was at the same time capable of processing a high transaction flow, we had to implement a number of unique solutions.

Given the implementation of quilt technology, Dynemix features two types of minting nodes: a full node and a master-node. Although master-nodes provide a number of benefits to the system, they cannot directly affect state transition, which is why decentralization is considered at the full-node layer.

Full nodes store only the current state and information about recent state changes. This does not require much storage space even under an intense load. A very important achievement is maintaining the storage and traffic overhead dependent only on the current system load, regardless of the time that has passed since launch (i.e. the growth of the blockchain does not affect the overhead). Full nodes are not obliged to store the entire blockchain and can securely synchronize with the system any moment without the need to download a complete set of missed blocks.

Sharding helps optimize the validation process on both the blockchain and quilt layers, and, according to our estimates, a mid-class home PC will be capable of participating even after the system significantly scales up.

Master-nodes store the complete blockchain with all transactions included. Running a master-node requires much more resources, but this mode is expected to be employed for professional purposes by services such as block explorers, exchanges etc. Though they store a lot of useful data, master-nodes are not crucial, and, even in case of their complete shutdown, the robustness of the system will not be threatened (although the additional line of defense will become unavailable).

Considering that full nodes are the main type of nodes that ensure system availability and resilience, we can state that Dynemix provides an exceptional level of decentralization.

3) Development

Our system features no obstacles to launching forks. Major stakeholders will not be able to compromise undesired forks; therefore, from this point of view, Dynemix's decentralization level is very high and equal to that of other open PoS systems.

4) Overall level

We can conclude that we managed to greatly increase the decentralization of every component, thus creating – without any exaggeration – one of the most decentralized platforms to date. Moreover, we have put a lot of effort into ensuring that the achieved parameters will not degrade as the system scales up with popularity.

IX. Payment channels and smart contracts

1. Payment channel technology summary

One of the major flaws of the first blockchain implementations was the scalability issue. Since no solution was found that could allow this issue to be resolved on the blockchain layer without sacrificing security or decentralization, the developers turned to second-layer solutions.

Payment channels are an off-chain solution that can take the load off the main layer and allow fast transactions to be made without the direct use of a blockchain.

The most commonly known implementation of the payment-channel technology is Lightning Network. It is believed to be the solution to Bitcoin's block-capacity issue.

We strongly believe, however, that it is not. The reason is that Lightning Network is not a part of Bitcoin. It is a separate payment system that can use bitcoins as transferable units, while at the same time not actually being a part of the Bitcoin protocol, apart from its opening and closing points. Technically, each payment channel is a separate two-replica SMR system with specific liveness and safety properties.

For this reason, it simply cannot solve any Bitcoin architecture issues directly. It is the same as claiming that fiat cash can solve the scalability issue because anyone can exchange Bitcoins for fiat cash, perform any number of transactions he or she wishes and purchase Bitcoins again in the end.

We have already expressed our skeptical attitude toward payment channels throughout this white paper, and now it is time to explain why we do not consider this technology beneficial, as it is often presented.

Payment channels feature some technical restrictions that make their implementation very specific.

- Channels can be opened only between two accounts.
- Channels' capacity is limited by the amount of the opening transactions.
- Channels exist within a predefined amount of time.

How does this affect the user experience? In most cases, money is transferred one way (from a client to a merchant). Situations when back-and-forth transfers are needed are extremely rare and often can be resolved by debt-clearance solutions without the direct involvement of the payment system. Considering that a payment channel requires at least two transactions to be sent to the blockchain (opening and closing) from each party, it only makes sense to open a channel between Alice and Bob, when Alice certainly knows that she will need to make more than two token transfers to Bob in a short period of time.

Given that a payment channel's capacity is strictly limited by the amount agreed upon by the parties in the opening transaction, the range of application of this technology seems quite limited. For now, we see that this can mostly be used for various kinds of repetitive and streaming services, in cases where it is convenient to pay for small portions of distributed content from a certain provider.

2. Payment channels network

To increase the capabilities of the technology, a relaying solution was introduced.

Suppose Alice wants to send tokens to Bob but does not have a direct open channel to Bob. At the same time, Alice has an open channel to Carol, and Carol has an open channel to Bob. Alice can use Carol as an intermediary to transfer tokens through the existing-channels route. Such a route can also be complex and involve multiple relays.

The routing system can unite all users in a single network and provide fast and secure off-chain token transfers, but, unfortunately, its design features certain limitations, which are the reason we are not optimistic about this technology.

Since, after opening a payment channel, the specified number of tokens is blocked in the blockchain until the moment the channel is closed, it is impractical to open payment channels just in case. This means that if Alice opens a channel with Carol, she pursues one of the following goals:

- She wants to transfer some tokens directly to Carol. In this case, using this payment channel as a relay for the token transfer $A \rightarrow C \rightarrow B$ is pointless, since it will reduce the capacity of channel AC, thus forcing Alice to open another channel to be able to settle her deal with Carol, when it would be easier for Alice just to open a new channel directly with Bob.

- She predicts that Carol will be willing to become an intermediary for token transfers to various recipients, possibly including Bob or someone who has a channel to Bob.

From Carol's point of view, the situation looks quite similar. If she opens a channel with Bob, then she pursues one of the following goals:

- She wants to transfer some tokens directly to Bob. In this case, using this payment channel as a relay for the token transfer $A \rightarrow C \rightarrow B$ is pointless, since it will reduce the capacity of channel CB, thus forcing Carol to open another channel to be able to settle her deal with Bob.
- She predicts that someone will be interested in transferring tokens to Bob and wishes to become a relay for that transfer.

This shows us that being a relay is not something that any user will be willing to do. Becoming a relay only makes sense if the user possesses a significant number of tokens in the blockchain, which she is ready to block, and she can ensure that enough users will be willing to use her services.

These obstacles make relaying a purely professional activity and transform a payment channel network into something resembling a banking system endowed with the following properties:

a) **Relays take fees for their services.**

For Carol, being a relay requires possessing and blocking the number of tokens equal to the overall capacity of the channels opened simultaneously to all other relays and/or clients. The only reason Carol may wish to bear this burden is to get an economic return.

A very important condition is whether the channel is opened in a PoW or a PoS blockchain. In PoW systems, users cannot earn tokens for staking, so hodlers may be initially interested in becoming relays.

PoS is a totally different matter – since being a relay requires the blocking of tokens, Carol will be willing to become a relay only if she can charge fees comparable to a possible income from block production. Otherwise, she will prefer to use the same number of tokens as stakes for minting.

If a PoW system supports Turing-complete smart contracts, it may become a timebomb for the payment channel network built on top of that system. Certain smart contracts may enable the placing of higher-yielding deposits (e.g. DAOs, crypto-collateralized stablecoins), which will incentivize hodlers either to raise fees in the payment channel network to ensure a level of return equal to the smart contract's yield or to relocate deposits, thus drastically lowering the network's bandwidth.

At the same time, the route between Alice and Bob may include multiple relays, each of which will charge its own fees, thereby possibly making the total fee higher than a fee for a direct on-chain transaction.

b) **Relays are points of failure.**

A payment-channels network is not a peer network, as most blockchains (including Dynemix) are.

Its overlay topology may be roughly described as hybrid star-mesh. Multiple clients are connected to a certain central relay, which forms a star topology. Then relays are connected to each other through a partially connected mesh network.

Such topology means that in the case of Carol's failure all her clients will be fully disconnected from the network and their tokens will be blocked in the blockchain until the channels' TTLs expire or Carol goes back online.

Clients cannot simply solve this problem by connecting to multiple relays, since each payment channel requires the blocking of tokens in the blockchain; therefore, clients will have to block a separate share of tokens for each connection.

c) **Relays have censorship abilities.**

Though Carol cannot steal tokens from Alice or Bob, she has strong censorship abilities, since she can refuse to fulfill her part of the deal by her own will, and Alice's and/or Bob's tokens in the channel will be blocked until the channel expires. Due to the limitations of the payment-channels technology, such actions cannot be penalized.

In most cases, these opportunities do not pose a serious threat, since such behavior does not provide any benefit to Carol. Moreover, she is economically incentivized to behave correctly in order to attract more customers to her mediation services.

Carol may be vulnerable to pressure, however. The more clients Carol has and the more she risks losing, the more cooperative she will become. Considering that Carol's node is a point of failure for all her clients, we may conclude that a payment channel network is highly vulnerable to censorship.

These features make the user experience with payment channels drastically different from the first-layer blockchain. A payment-channel network lacks decentralization, fault tolerance and censorship resilience – very important features of blockchain systems. At the same time, most common users do not recognize the difference between the first and the second layer and consider Lightning Network a part of Bitcoin.

This is why we do not consider the payment channel technology a good solution to the scalability issue. The technology is not bad in itself, but the misrepresentation of its nature confuses cryptocurrency users, who for the most part do not have sufficient competence to understand all the flaws of the mentioned solution.

In defense of payment channels, we can say that, for now, they are supposed to be used mostly for microtransactions, and it seems to be the best way to use them, since payment channels can provide the necessary speed for slow blockchains like Bitcoin. At the same time, the flaws mentioned above become less crucial when the stakes are low.

3. Atomic swaps and cross-chain transactions

HTLC (the underlying technology that powers payment-channel networks) can be also used for atomic cross-chain swaps. An exchange of different cryptocurrencies can be performed directly by two parties by opening payment channels in both blockchains and closing them after the parties fulfill their obligations or the TTL expires.

The proposed solutions for atomic swaps via HTLC seem generally reasonable, as they solve the problem without intermediaries and thus keep decentralization on the level of the first layer.

Unlike simple swaps, the problem of cross-chain transactions between arbitrary blockchains requires a more complex solution.

To date, there are several concepts for resolving the matter by involving second-layer intermediaries or building a top layer above the blockchains that is controlled by a consensus of validators, thus creating a heterogeneous multi-chain (e.g. Polkadot). This should become a decentralized solution for cross-chain transactions.

The problem is that the currently proposed protocols are not as decentralized as we would wish them to be. For example, Polkadot uses Delegated Proof of Stake as a consensus model for the top layer (Relay Chain). DPoS indeed can be faster than most PoS alternatives, but we are quite skeptical about the level of decentralization that can be achieved with such a design.

Considering that joining such systems requires modification to the blockchain protocol and granting of authority to the top-layer validators, we do not find solutions of this kind appropriate for Dynemix, and the system will likely not go beyond cross-chain swaps. The final decision will be up to the community.

4. Payment channels in Dynemix

Taking into account the above, atomic swaps via HTLC seem to be a viable option. Therefore, in the case that this technology becomes widely adopted, adding the support of HTLC to Dynemix to enable atomic swaps seems an option worth considering. The final decision will depend on the attitude of the community.

As for the payment channels themselves, there is not much use for this technology for Dynemix, since our system does not suffer from the scalability issue. Considering that payment channels form a separate network with its own rules (followed by the drawbacks mentioned above) and that recognizing the payment-channel network from the blockchain layer may be troublesome for most users, we would advise avoiding the use of this technology unless it becomes clear that it provides some really significant benefits (like the mentioned atomic swaps in the case of their mass adoption or the possibility of implementing off-chain smart contracts).

It is also worth mentioning that a full-scale payment-channel network cannot be formed on top of Dynemix for economic reasons. Since Dynemix features free transactions, there are no economic prerequisites to become a relay – if a relay charges commission for its mediation services, users will prefer sending transactions directly to the blockchain free of charge. At the same time, it makes no sense to perform the relay functions in a payment-channel network without receiving a reward, since users could engage in minting instead. The situation may change, though, if Dynemix begins to support off-chain smart contracts, which are not supported on-chain.

5. Smart contracts in Dynemix

As mentioned, implementing virtual machine technology can negatively affect the platform's stability and scalability, as well as degrade certain features that allow Dynemix to be a breakthrough payment platform.

For these reasons, we will not include any smart-contract support in the first version of the system.

If the system, however, shows a performance reserve after being tested under substantial loads (with at least 10 million active users), adding some limited smart-contract support may be considered. This should most likely not include Turing-complete code support, but rather specific templates with a limited set of variables that can actually benefit the cryptocurrency as a means of payment and at the same time not completely turn it into a decentralized virtual machine.

As mentioned, adding certain types of smart contracts can allow not only the implementing of scripted business scenarios, but also the possibility of creating payment channels on top of Dynemix and using a variety of off-chain smart contracts with the help of DLC or any other possible techniques that can be developed in the future. Of course, such contracts will have limited functionality and will not bear the same properties as smart contracts operated on the blockchain layer, due to the flaws of second-layer solutions, but nevertheless it is a viable option for implementing smart contracts without increasing the load on the blockchain.

We should note that such a decision would require a hard fork, so the possibility of implementing smart contracts will depend on the community's attitude.

X. Personal data, censorship and interaction with authorities

1. Unbiased content filtering

We stand for the protection of the substantive human rights, with freedom of expression among them. Unfortunately, human rights can be also exercised solely for malicious purposes. This is why freedom of speech is usually not considered absolute and is subjected to certain limitations.

The problem is that, once we appoint a censor, we cannot be sure that such limitations will be applied objectively and impartially. In real life, it often happens that governments use their censorship abilities to oppress any opposing opinions and deprive people of their ability to express their disapproval of government actions.

Since we are utilizing a blockhain protocol with embedded strong censorship resilience, it would be unwise not to use the emerging opportunities for censorship-free interaction. At the same time, we should secure certain ways of preventing users from abusing this opportunity for malicious activities.

To solve both problems at the same time, we decided to apply limited content filtering via guardian oracles, which will have the following specs:

a) **Blacklists of banned users will be administrated by a specially appointed entity or by any third parties.**

   By default, when a user downloads any version of the Liberdyne messenger, it will contain an active blacklist of user accounts that have been accused by an appointed entity of breaking the rules. In the early stages, the blacklist will be fed by an oracle controlled by developers. Later, the functions of maintaining the oracle may be transferred to a specially created organization.

   There will also be an opportunity to apply any other blacklists, including those administrated by third parties through personal preferences, if the user wishes. For example, there can be organizations monitoring certain types of undesired content or parental-control services that the user wishes to apply.

   It will be also possible to create decentralized oracles that will be controlled by cumulative voting or any other possible types of oracles that the community may come up with.

b) **Filtering exists only on the messaging-application layer.**

   Filtering will be applied only toward the messaging functions; blacklisted users will not be able to send messages, calls, files or requests to be added to contact lists or perform any other messaging functions toward other users.

   At the same time, no censorship of any kind should be ever applied on the blockchain layer, since it is a straightforward violation of decentralization.

   If necessary, it will be possible to create a decentralized oracle that will manage censorship on the blockchain layer (i.e. provide a set of default rules for the Guess My Block game). We are uncertain whether this approach is appropriate, however. The final decision is up to the community.

c) **Filtering can be deactivated by the user at any time.**

   Once the user decides that he does not need this kind of protection, he can easily disable filtering through the app's preferences and get a censorship-free messaging app.

> We cannot guarantee that such an opportunity will be available for all platforms, since it may violate the platform administrator's policy.
>
> The function of enabling/disabling blacklists will be assuredly available on the *nix, Windows and Android versions, since these platforms do not restrict the installation of apps downloaded from an arbitrary source.
>
> As for the versions distributed through the App Store and Google Play, this opportunity will solely depend on Apple's and Google's attitude toward it. In the case that we get any claims form Apple or Google, we may have to remove the blacklist-disabling feature from the relevant version.

The combination of these features ensures an unbiased approach to filtering and at the same time provides sufficient protection from malicious content distribution. Even If the developers (or other entities controlling the default guardian oracle) try to apply any prejudiced filtering rules, users are free to choose another filtering solution, which renders such attempts inefficient.

In general, filtering will be mostly used to protect users from spam and certain kinds of undesired content, but in the case that the user wishes to access any content, he can easily disable censorship. We believe that it is up to the user to decide how he wants to use the possibilities provided by the app, unless his actions violate another user's rights.

We do not support any darknet activities, but we also understand that it is impossible to fully restrict them. For example, we can restrict users from disabling the blacklist, but, since it works on the application layer and not on the protocol layer, any third party can modify the code (as we go open source) and create an alternative client that has no censorship. Therefore, such actions are pointless. On the other hand, if we try to apply censorship to the protocol layer, this would instantly ruin decentralization, which is unacceptable.

2. Liberdyne and personal data

Many countries have adopted comprehensive data-protection laws. The most commonly known is GDPR, which regulates the processing of personal data within the European Union. Since the problem has been widely discussed, we also took it into account while building the system architecture.

Our approach is simple – due to its distributed architecture, the Liberdyne messenger will not collect or send any user data, either to us or to any other parties, unless it is required by the transport protocols (for example, if the user enables relay chains for anonymity purposes, all his or her correspondence will be routed through a chain of nodes in encrypted form). This can be easily verified by anyone, since the source code of the app will be publicly available. Our access to user data is limited to the information held in the public domain; therefore, we do not have to comply with any data-protection regulations.

There may be, however, certain situations when users provide us with their personal data by choice, for example, while contacting the support or abuse departments. In such cases, we will process the personal data according to the regulations, and all of the collected data will be instantly deleted after the matter of the appeal is resolved (unless the regulations require temporary storage).

3. Interaction with authorities

Since we do not run any servers that process user data or participate in the system operations in any other way, apart from running our own peer nodes, we cannot collect any user-generated content that may be of interest to governments, corporations or other parties.

If any information about the user's personal data, communications, interactions with other users etc. is requested by authorities, we will not be technically able to satisfy such a request.

This may cause pressure from the authorities of certain states, where developers of a communication app are obliged to implement means of obtaining any user's information (including transmitted content, like private messages, files etc.) and providing it to the authorities upon request.

We believe that such laws go far beyond reasonable human-rights limitations, and we do not support this paradigm. We have intentionally developed an architecture that fully prevents any possibility of imperceptible user data obtainment, either by ourselves or by any third parties.

Our approach is fully consistent with the ideas outlined in the International Covenant on Civil and Political Rights, the UNGA Resolution 68/167 and the subsequent report of the United Nations High Commissioner for Human Rights A/HRC/27/37.

Nevertheless, we understand that our product's release may cause a negative reaction from certain autocratic governments that intend to fully control the private lives of their citizens, and that we will face attempts to block Liberdyne in a number of countries. Considering the current political situation, these will most likely be North Korea, Russia, China and Iran.

These attempts will most likely fail, considering that Liberdyne is designed to resist blocking attempts much better than any centralized messaging app possibly can.

On the other hand, the majority of democratic regimes around the world stand for the protection of substantive human rights and the freedom of expression among them, hence, considering that we apply reasonable limited censorship to restrain the abuse of those rights, we expect to avoid confrontations with the authorities of most countries.

XI. Open source

From the very start, the crypto-community has relied on certain principles that form the basis of crypto-philosophy. One of these principles is to make cryptocurrency software fully free and open-source to ensure the possibility of the decentralized and transparent development of the platform.

To prove the proclaimed security of a messenger app, its source code should also be open to public examination and independent audit.

Considering the importance of making both components of our app open, we intend to make Liberdyne fully free and open source, distributed under the GNU GPL (other open licenses may also be considered).

If we remove copyright straight from the start, however, we may provoke the creation of numerous scam and copycat projects, which will try to confuse users. Without any legal protection, it will be challenging to deal with such behavior, and this can cause a lot of problems both for us and for users, who will find it difficult to distinguish the original Liberdyne/Dynemix.

At the same time, we do not want to register any patents for our solutions, for we intend eventually to make the platform fully free and open, and we should not obstruct the process with our own actions.

To solve this matter, we have decided to distribute the software under different licenses during two periods.

a) **The proprietary period** – from the start of the public testing to a short period after the mainnet launch.

   From the first publication of the software, it will be distributed with an open source code but under a proprietary license, which will cover the source code by copyright and restrict

modification of the code. This is especially important for the test period, as the creation of modified versions and forks by users may cause turmoil and negatively affect the development process.

b) **The free period** – from a short period after the mainnet launch and onward.

Shortly after the mainnet launch, Liberdyne will be distributed under the GNU GPL, thus removing any copyright protection and making the software fully open and free.

XII. Competition

It should be noted that the project actually has no direct competitors since the information about other cryptocurrency platforms created in a symbiotic relationship with the basic application for distribution has not been published even in the form of a project, and we simply have nothing to directly compare the Liberdyne/Dynemix project with.

Nevertheless, as separate components, Liberdyne and Dynemix can be compared to other solutions on the market.

1. Dynemix and other blockchain platforms

As a cryptocurrency designed especially for everyday use as a payment tool, Dynemix does not directly compete with any existing project but rather forms its own niche. We believe that blockchain platforms should be classified as follows:

a) **Digital gold (Bitcoin and similar altcoins)**

As we already explained, despite having been called an electronic cash system, Bitcoin actually cannot perform cash functions and compete with conventional centralized payment platforms.

This happened due to the technical limitations of the platform and its inappropriate economic model. Other altcoins that were not designed to be smart-contract platforms (such as Litecoin, Monero etc.) mostly share the same properties.

These blockchains can serve as a hedging tool, investment asset or tool for settlements between big players in the market, but not as a universal medium of exchange on the consumer level.

b) **Decentralized virtual machine (Ethereum and similar altcoins)**

This group consists of platforms that allow the execution of Turing-complete decentralized scripts. Despite such systems' being theoretically capable of performing the same functions as standard payment platforms, the additional complexity negatively affects their payment features and makes it difficult for such systems to compete with payment-focused projects on equal footing.

Since, after the Ethereum release, most developers completely abandoned Bitcoin-like concepts and concentrated on trying to create an improved version of a decentralized virtual machine, almost all recent projects can be assigned to this group.

c) **Decentralized currency (Dynemix)**

Unlike projects belonging to the previous categories, Dynemix is designed especially to be a universal medium of exchange for a wide audience. As mentioned, performing this function

requires certain technical, economic and social features that other platforms fail to deliver. This is why Dynemix forms a separate group according to our classification.

During Dynemix's development, we came across several projects with the same proclaimed goal, but we still have not seen any architecture that can allow this goal to be achieved.

For example, in the alleged TON project white paper, the authors claim they intend to develop a mass-market cryptocurrency (basically what Dynemix is), but, in the subsequent technical paper, they actually describe a kind of centralized heterogeneous multichain smart-contract platform of unclear purpose based on Polkadot architecture, which is obviously not suitable for performing the proclaimed functions and cannot compete technically with Dynemix.

We believe, however, that after our project's release, some other teams interested in our concept may present their own vision, and Dynemix will not feel lonely in this category for long.

2. Liberdyne and other decentralized messengers

Despite there being several messengers that use blockchain technology, Liberdyne is unique due to its development in conjunction with Dynemix.

As a rule, blockchain platforms are created separately as a tool for the development of a wide range of applications, and decentralized messengers are created by other developers as applications on top of existing platforms, although there are a few exceptions. For example, Adamant Messenger uses its own blockchain but is actually based on Lisk, and its main purpose is message delivery; hence, it differs from our concept.

On the other hand, the Dynemix/Liberdyne project was developed as a whole, and both components are interdependent and mutually beneficial. This fact puts Liberdyne ahead of other P2P messengers since Liberdyne features a well-designed reward system that encourages users 'interest in supporting system operations without relying solely on their altruistic intentions.

Another noteworthy feature of Liberdyne is its optimal use of blockchain technology. After the cryptocurrency boom in 2017, a number of developers introduced concepts of decentralized messengers based on blockchain technology. The problem was that many of these projects used blockchain technology for the sake of technology, not intending to create an optimized solution that could help improve the user experience and create an ultimate product in the first place.

As a result, to date no one has been able to present a product capable of competing with popular centralized solutions.

On the other hand, Liberdyne uses blockchain technology only to the extent that allows it to achieve the proclaimed goals and present a product that can withstand significant loads, providing a user experience that is not substantially inferior to centralized solutions (although we have to admit that we can hardly manage to reach the same exact level of speed and reliability as popular centralized solutions due to the limitations of P2P architecture).

XIII. Marketing strategy

We understand that, though we are developing a breakthrough project that may let the whole industry take a huge step forward, only those few who are deeply involved in cryptocurrency technology will be able to fully evaluate Dynemix's potential.

Since our main goal is not just to create a new advanced platform for the cryptocommunity in its current state but to spread blockchain technology all over the world via the creation of the first

mass-market product, it is absolutely clear that about 99% of our target audience has a very limited basic understanding of the technology. While we can convince some portion of the cryptocommunity members of Dynemix's strong potential simply by explaining all the advantages of the platform, it will not work with people who are not much into blockchain technology.

There is no way we can capture a major audience's attention long enough to explain all features of the Dynemix platform and benefits it can bring to people's everyday lives, which is why we need to introduce another approach that will help us raise interest in the platform even among those who do not plan to join the cryptocommunity.

To accomplish this, we need some simple, and at the same time striking, "killer features" that can be easily explained and understood, thus driving public attention to the project by themselves. Fortunately, we have a couple of those.

1. Liberdyne – the next-generation decentralized private messenger

As mentioned, the revelations of Edward Snowden, according to which the US special services had been actively collecting people's private information on a global scale, led to the rise of discussions about creating a secure means of communication that could guarantee users' privacy.

To solve this problem, the developers of instant messengers implemented end2end encryption. Telegram was the first popular app to do so, and it was its major marketing feature. In fact, it was the main distinguishing feature of the Telegram messenger back at its launch, while, from the point of view of functionality, it was inferior to its competitors (it did not even feature voice calls). Using that single feature as the basis of the app's marketing helped Telegram rapidly gain a significant userbase and forced other competitors to quickly implement end2end encryption as well.

This example shows us that the question of user privacy can be a strong basis for a marketing campaign. Even though end2end encryption has not solved the problem of user privacy, since users still have to trust the developers, who operate servers through which all user data passes (which opens up an opportunity for the meta-data analysis), it helped promote a product to the level of its competitors, which were years ahead of it at the time.

Liberdyne makes a much larger step toward user privacy – it is a next-generation product that works fully P2P without any servers involved. In combination with the peer relaying technology (SATAN), which provides a whole new level of privacy and security to every user, it can finally give users confidence that their communication is reliably secure.

Due to the excellent censorship resilience of the Dynemix blockchain, we managed to implement a new content filtering solution into Liberdyne, which can allow for decentralized unbiased user-selectable filtering. This approach will put an end to biased censorship, which has been applied by major social platforms during the past years.

Given that Liberdyne is not just offering a slight protocol improvement but is a generation ahead of all major centralized instant messaging software, while at the same time keeping most current functionality and parameters available, it may provide strong incentives for users to download and try it.

In addition, we should note that the major centralized messengers will not be able to offer the same quickly, since it requires a complete architecture redesign. That fact should keep the project ahead of any competitors for a long time.

2. Liberdyne – the messenger that pays you to use it

By default, the application will be configured to perform a number of functions to support the system (namely minting, DOG, SATAN relays, and storage nodes), for which each user will automatically receive a small reward, becoming a sort of "miner." At the same time, the functionality

of the mobile versions will be configured so that the user will not feel that there is a significant waste of resources (i.e. battery and Internet traffic). If desired, through the settings, the user can both increase the amount of resources spent (thereby increasing the reward) or reduce or even disable the feature entirely.

As a result, users will receive a reward in the Dynemix cryptocurrency for the mere act of using the application (as “usage” includes system support, like in all P2P systems). At the same time, the algorithm of reward distribution will be such that the fewer users who are online, the greater the reward each of them will receive. According to our idea, this will encourage users to install the application as quickly as possible in pursuit of a larger reward, which should lead to an explosive growth of the userbase. When the userbase becomes large enough, people will already be interested in using the application for reasons other than receiving a reward.

Getting a small reward and having the ability to spend it should incentivize a huge number of new users to become involved in the cryptocurrency world, allowing them to overcome the entry threshold. Thus, we will gain a loyal userbase that has experience using the Dynemix cryptocurrency.

Having learned to use the cryptocurrency on a virtually free basis and realizing its advantages, people will be able to gradually exchange their fiat money for it and come up with other scenarios for its use by themselves (for example, for payments between users).

Businesses may also be interested in using Dynemix for payments, given the size of the loyal userbase that Liberdyne may obtain, and the possibility of using Liberdyne as a distribution channel that can repeatedly exceed the capabilities of any other cryptocurrency platform.

The inclusion of more users in the Dynemix economy should stabilize its rate and make it a convenient tool for any kind of payment.

Thus, through the use of this simple slogan and the concept behind it, the project has the potential to take a leading position in the cryptocurrency market.

This feature will also help overcome users’ initial reluctance to use a new P2P messenger (which means a lack of available contacts within the app in the beginning), as people will be incentivized by receiving more rewards with a smaller userbase.

Combining these two “killer features,” we can conduct a marketing campaign that will hit both the instant-messaging market and the cryptocurrency market, reaching the largest potential audience and helping the platform to gain a significant userbase. We expect that this approach has the potential to secure leading positions for the project in both markets.

XIV. Monetization and revenue

1. Monetizing Liberdyne

We do not intend to monetize the Liberdyne messenger directly. Liberdyne is conceived as a basic platform for communication and payments and as the foundation of the entire Dynemix ecosystem, which is why we intend to keep it completely non-commercial. All paid services integrated into Liberdyne (e.g. SATAN, cloud storage) will be directly P2P and will bring no revenue to the developers.

Unlike most products with a centralized architecture (e.g. WhatsApp, WeChat), in our case, the cost of supporting the application will be much less.

This is due to the fact that the operation of the system is secured by the users themselves (with P2P architecture), which eliminates the need to maintain powerful data centers to process huge flows of information.

When it comes to the technical support of the infrastructure, developers play the same role as any other users and only during the starting period will the involvement of the developers be necessary. According to our estimates, we will need to run about 100 master-nodes to guarantee stable system performance.

After attracting a sufficient number of users, if additional income is necessary to support the system, we may consider ways of monetization:

- Adding additional services to the application that will be paid for by the users in dynes.
- Charging a fee to service providers for their content distributed via the messenger.

We see monetization, however, as a last resort and hope that we will be able to avoid it.

2. Other projects on the Dynemix platform

Most developers of cryptocurrency platforms are not involved in building the ecosystem around the platform. As a rule, they create basic state-transition software and delegate everything else to third parties.

We may take another approach, and we do not exclude the possibility of participating in the development of the platform's ecosystem, which refers to both the direct development of different services and the support of third-party projects. At this stage, however, we cannot reliably confirm our intentions, as it is not currently clear how large a set resources we will possess and how our priorities will be set.

XV. Dynemix and scams

Dynemix is not a scam.

It is highly likely, however, that some unscrupulous actors may try to take advantage of the fact that we do not conduct a public token distribution before the mainnet launch. We highly recommend staying extremely cautious toward any announcements and offers of tokens committed on our behalf that are not directly confirmed on www.liberdyne.com. The project has no other official domain names, and no tokens will be available for distribution without prior notice on the website.

If someone offers you dynes that were allegedly placed during the closed sale or obtained directly from a member of the team, be aware that it is undoubtedly a fraud attempt.