Platform white paper

Version: 0.11 Date: 2019-05-20

- Platform white paper
  - Executive Summary
    - Motivation
    - Technical design & features
    - Uses
  - Design rationale
    - Overview of problems with existing platforms
    - Blockchain as a database
    - Relational model
    - First-class decentralized applications
    - Programming model
    - Consensus & nodes
      - Model overview
      - Sybil control mechanism
      - Consensus
      - Node compensation
    - Miscellaneous features
  - Decentralized applications
    - Transparent apps
    - Token model
    - The role of Chromia
      - Not controlled by a single entity.
      - Controlled by the community of users.
      - Cannot be shut down.
      - Censorship-resistant.
      - Transparent.
      - Privacy.
      - Highly available.
    - Decentralization quality
  - Platform architecture

    - Postchain
    - Chains
        - System chains:
    - Node implementation
    - Interaction with other blockchains
    - Components
- Governance
    - Chromia system governance
        - Initial centralization
        - Rejected alternatives
            - Stake / coin voting
            - No formal governance
            - Unique users
    - Application governance
- Uses
    - Tokens
    - Games
    - Business uses
- Tokens and incentives
    - Fees
        - Application fee models
        - Hosting fees
        - Node incentives
        - Node stakes
    - Token use in games
    - Chroma token economics
    - System accounts
        - Public good account
    - Token distribution
        - Promotional token fund
- Decentralization
    - Centralization necessary at start
    - Decentralization through a diverse set of providers
        - Bitcoin
        - DPoS
        - Ethereum
        - Chromia
    - Number of full nodes
- Security
    - Blockchain
    - Node security
    - Governance security

Light client security
Dapp client and wallet security

Executive Summary

Chromia is a new blockchain platform for decentralized applications, conceived in response to the shortcomings of existing platforms and designed to enable a new generation of dapps to scale beyond what is currently possible. Chromia was previously named Chromapolis.

Motivation

While platforms such as Ethereum allow any kind of application to be implemented *in theory*, in practice they have many limitations: bad user experience, high fees, frustrating developer experience, poor security. This prevents decentralized apps (dapps) from going mainstream.

We believe that to address these problems properly we need to seriously rethink the blockchain architecture and programming model with the needs of decentralized applications in mind. Our priorities are to:

- Allow dapps to scale to millions of users.
- Improve the user experience of dapps to achieve parity with centralized applications.
- Allow developers to build secure applications with using familiar paradigms.

Technical design & features

We believe that a blockchain serves the role of a **shared database** within a decentralized application ecosystem: it stores application data and makes sure that data additions, updates and transformations are authorized and consistent with the application's rules. For this reason, Chromia is designed and optimized to serve the role of a shared database in the best way possible. It is implemented using the existing Postchain[1] framework developed by ChromaWay, and It features:

- A relational model[2]: Blockchain data and application state are stored in a relational database. This model is considered to be best in class in terms of flexibility, versatility and consistency.
- A relational programming language: Chromia dapp backends are written in a specialized language which is deeply integrated with the relational model. This model increases programmer productivity and ensures application consistency.
- Horizontal scaling: Each dapp has its own blockchain(s). Because each blockchain is run by a subset of nodes, it is possible to increase total throughput by increasing the number of nodes.

[1] https://chromaway.com/products/postchain/

[2] Codd, E.F (1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM. Classics. 13 (6): 377–87; https://dl.acm.org/citation.cfm?doid=362384.362685

- Rich indexing and querying: Dapps can quickly retrieve information they need directly from nodes running the application. Dapp blockchain logic can perform complex queries without severe performance degradation.
- High I/O throughput: data queries and updates are delegated to a heavily optimized relational database, allowing dapps to perform a large number of queries and data update operations.
- PBFT[3]-style consensus: Transactions can be confirmed within seconds.
- First-class dapps: Dapps do not arise from "smart contracts" in Chromia, but are considered first-class entities. Chromia gives dapp developers a high degree of flexibility and control.
- Dapp level provisioning: allocating resources to dapps rather than contracts gives developers the freedom to create their own fee and resource use policies.

Chromia offers the same level of openness, transparency and decentralization as other public blockchains. In Chromia *miners* are replaced with *providers.* Providers own[4] nodes which produce blocks. It has been suggested that the four largest mining pools of both Bitcoin[5] and Ethereum[6] could exert significant control over those networks if they colluded. We aim to ensure that the minimum number of node providers whose collusion would be required to exert such control on Chromia exceeds this number significantly. It can therefore be said that the Chromia model does not tend towards centralization any more than the oldest and most trusted public blockchains.

Chromia' PBFT-style consensus is further hardened by anchoring[7] Chromia chains to a Proof of Work (PoW) blockchain, likely Bitcoin or Ethereum. This ensures that finality, the guarantee that confirmed transactions cannot be changed, is at least as strong as that of the chosen anchoring chain. To alter the history of an anchored portion of Chromia block history it would be necessary to combine PoW blockchain reorganization with a malicious collusion of a sufficient number of Chromia nodes. The likelihood of any attacker having the resources to successfully mount such an attack is extremely low.

Uses

Chromia is a general-purpose platform which is suitable for almost all kinds of dapps. It is particularly well suited to cases requiring high I/O capacity or involving management of complex data sets. Massively multiplayer online games (MMOGs) are an example of such a case.
Blockchain gaming is becoming increasingly popular, but MMOGs are currently out of reach because no existing blockchain platform can support them. Chromia is capable of hosting entire

[3] Castro, M.; Liskov, B. (2002). "Practical Byzantine Fault Tolerance and Proactive Recovery". ACM Transactions on Computer Systems. Association for Computing Machinery. 20 (4): 398–461. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.127.6130

[4] We say "own" rather than control, because the incentive system for providers is economic in character: they own Chromia resources, and derive profit from them. There is some tension between the notions of ownership and of control of those resources, as the control of Chromia resources is arguably a more relevant criterion for assessing the diversity of the provider pool.

[5] https://blockchain.info/pools

[6] https://www.etherchain.org/charts/topMiners

[7] We originally described Anchoring as "side-chains" https://bitcointalk.org/index.php?topic=313347; a more formal discussion of anchoring can be found in the BitFury white paper "On Blockchain Auditability". https://bitfury.com/content/downloads/bitfury_white_paper_on_blockchain_auditability.pdf

game worlds in the blockchain, making sure that they evolve according to predetermined rules and ensuring that no one can cheat. We believe that implementing a MMOG will be the best way to showcase the capabilities of Chromia. MMOGs have a very demanding set of requirements; the capacity to run MMOGs implies that Chromia is suitable for demanding and complex dapps of all kinds.

Design rationale

Overview of problems with existing platforms

Ethereum was the first blockchain to offer a platform for decentralized application development. Many application prototypes were created, but developers faced the following issues:

- Limited capacity. Because network capacity is limited, and usage fees are proportional to load, transaction fees can be $1 or more for complex applications. This cost, typically paid for each interaction with an application, makes most applications too expensive to be practical.
- Prohibitively expensive I/O operations, for the same reason. For example, a contract cannot iterate through a list of users since the cost of this action would exceed the block gas limit. Thus developers have to jump through hoops to implement something as simple as an interest payment to a list of users.
- Poor data modelling tools and poor support for queries. Application developers have to resort to centralized indexing and caching layers, or using third party services which do not provide the same security guarantees as the base layer.
- Error-prone contract language which has resulted in many high-profile heists.
- No provision for contract upgrades at the platform level, this functionality has to be implemented as a separate layer which further increases complexity.
- Users are required to pay a fee for every interaction and confirmations are slow. This results in poor User Experience (UX).
- Poor light client support. Three years after beginning development efforts the Ethereum Foundation was still struggling to offer a production-quality light wallet[8].

Applications designed with a large audience in mind need to be flexible and responsive. They require a platform which empowers the developer to allocate resources in a way that suits their users. Even if Ethereum and other platforms currently in development tackle scalability issues, they will not be able to provide a sufficient degree of developer autonomy and will remain a somewhat hostile environment for dapps.

We believe that to properly address these problems we need to seriously rethink the blockchain architecture and programming model with the needs of decentralized applications in mind.

[8] Light client started being somewhat usable around February 2018.

Blockchain as a database

The main role of a blockchain in a decentralized application context is to manage data in a secure and consistent manner. Thus, a blockchain can be understood as a database, more specifically as a secure decentralized database. Another major role of a blockchain is prevention of double-spending, but this is a special case of data consistency constraints.

Blockchains which are optimized for payments, such as Bitcoin, can adopt highly specialized (and optimized) data models. But a platform designed for hosting diverse decentralized applications needs a general-purpose data model.

Most blockchain platforms nowadays use key-value data stores (examples: Ethereum, NEO, Fabric). This model is, in theory, complete, and enables the use of high-performance data stores such as LevelDB. However, this model is very low-level and requires application developers to implement core functionality like serialization and indexing, a daunting challenge.

Compounding this, blockchain platforms typically do not expose the full functionality of key-value stores, such as the ability to use arbitrary-sized keys and iterate through keys. For example, in the Ethereum Virtual Machine (EVM) all keys are 256-bit integers and iterating through stored keys is impossible. For these reasons, implementing proper indexed data access on the EVM is both difficult and inefficient.

Relational model

The relational model has been the gold standard for database management for the last five decades. Rooted in mathematics and logic, it is known to be able to model complex data in an efficient way. For this reason, and the reasons stated above, we consider the relational data model to be the lynchpin of our blockchain platform.

As decentralized applications deal with increasingly complex data structures, the advantages of the relational model become ever more apparent. Further, most software engineers are already familiar with it so they won’t have to learn new concepts in order to implement an application.

A relational model also allows us to leverage the power of SQL database management systems (DBMS) which have been optimized for decades. Instead of dapp code which traverses memory cells one by one, we can send a query to the DBMS and let it use its sophisticated query planning, data structures and caching capabilities to carry out the query as fast as possible.

Of course, the choice of data model is a trade off. The relational model might have the following disadvantages:

- Performance is hard to predict and depends on the query planner. This is not a significant disadvantage in the context of Chromia because each dapp will be run in an isolated manner; slow queries will affect only the dapp which performs them rather than the system as a whole.

- It is impossible to impose hard bounds on query execution time. Again, this is not a problem in Chromia because it affects only the performance of the application which issues slow queries.
- Parallelization of SQL databases is a complex area of active research. As far as we know, no blockchain platform offers 100% fully automatic parallelization on a massive scale. Thus there is no evidence that a relational model is worse than other models. In addition, we believe that the relational model will make logical sharding and sidechain mechanisms easier to implement.

First-class decentralized applications

In Ethereum all code lives in "contracts". It does not differentiate between individual wallet contracts and complex multi-user contracts: they all use the same resource-metering and programming model. An Ethereum-based dapp will use one or more contracts (possibly a contract for each user) and front-end components. In fact, many Ethereum applications make use of centralized caching, rendering their "decentralized" credentials somewhat dubious.

While this approach is quite elegant and can scale to different kinds of applications, it is very inconvenient for dapps designed for mass use. End-users have to pay for every interaction with their dapp, in proportion to the computational and storage resources required for their transaction. In other words, Ethereum doesn't give decentralized applications the flexibility to manage resources themselves. For example, a "freemium" business model is outright impossible. This creates a barrier for decentralized application adoption: most users are not ready to pay for every single click.[9]

Chromia solves this issue by provisioning resources on the decentralized application level:

- Each dapp has its own blockchain (sidechain)
- Fees (collected to maintain nodes) are paid by the dapp as a whole, not by end-users directly

Consequently, dapps are free to implement their own resource management policies, which can be aligned with economic rather than technical needs.

Every blockchain needs an anti-spam mechanism, but this mechanism doesn't have to be tied to fees. For example, a dapp might allow only 1 action from a user each 15 seconds, thus a single user won't be able to spam the blockchain with billions of transactions. A dapp can also mitigate Sybil attacks through limiting new user registration to some reasonable rate and/or requiring invitation or a deposit.

In this model, we do not need to measure the resources used by each operation. Instead, we provision resources to the application as a whole: each dapp blockchain will run on a specific set of nodes and typically will have its own dedicated CPU thread. If a dapp needs more than one execution thread, it can consist of multiple shards each of which will be a sidechain.

[9] Indeed, the fact that the cost for the user increases as more people join the network (greater network congestion -> higher fees) is totally at odds with the economies of scale upon which computing technologies thrive.

This removes resource metering overhead (we no longer care how many instructions were executed, as an application cannot use more resources than it was given) allowing dapps to perform faster and scale better.

Besides scheduling, having dapps as first-class citizens on the platform allows token economics to be integrated with the fee model, i.e. fees are taken from profits “earned” by an application. It also supports mechanisms for governance and dapp updates which are built-in to the platform. These features are discussed in more detail later in this paper.

Programming model

The Postchain framework on which Chromia is based allows us to use existing open source SQL database software (specifically, PostgreSQL) to implement data store and query capabilities. However, we cannot allow dapps to perform arbitrary SQL queries as said queries might be unsafe, ambiguous or lead to excessive resource use.

Most dapp blockchain platforms use virtual machines of various kinds. But a traditional virtual machine architecture doesn’t work very well with the Chromia relational data model, as we need a way to encode queries as well as operations. For this reason, we are taking a more language-centric approach: a new language called Rell (**Rel**ational **l**anguage) will be used for dapp programming. This language allows programmers to describe the data model/schema, queries, and procedural application code.

Rell code is compiled to an intermediate binary format which can be understood as code for a specialized virtual machine. Chromia nodes will then translate queries contained in this code into SQL (while making sure this translation is safe) and execute code as needed using an interpreter or compiler.

Rell will have the following features:

- Type safety / static type checks. It’s very important to catch programming errors at the compilation stage to prevent financial losses. Rell will be much more type-safe than SQL, and it will make sure that types returned by queries match types used in procedural code.
- Safety-optimized. Arithmetic operations are safe right out of the box, programmers do not need to worry about overflows. Authorization checks are explicitly required.
- Concise, expressive and convenient. Many developers dislike SQL because it is highly verbose. Rell doesn’t bother developers with details which can be derived automatically. As a data definition language, Rell is up to 7x more compact than SQL.
- Allows meta-programming. We do not want application developers to implement the basics from scratch for every dapp. Rell will allow functionality to be bundled as templates.

Our research indicated that no existing language or environment has this feature set, and thus development of a new language was absolutely necessary.

We designed Rell in such a way that it is easy to learn for programmers:

- Programmers can use relational programming idioms they are already familiar with. However, they don't have to go out of their way to express everything through relational algebra: Rell can seamlessly merge relational constructs with procedural programming.
- The language is deliberately similar to modern programming languages like JavaScript and Kotlin. A familiar language is easier to adapt to, and our internal tests show that programmers can become proficient in Rell in matter of days. In contrast, the ALGOL-style syntax of PL/SQL generally feels unintuitive to modern developers.

The Ethereum programming model is typically described as very error-prone. Bugs in Ethereum smart contracts have resulted in losses totalling hundreds of millions of dollars[10]. In Chromia, we aim to eliminate most common sources of problems through a better programming model (no weird interactions between different smart contracts as in the DAO case[11][12]) and safer languages.

Since Ethereum code is immutable, it is often impossible for a developer to fix her dapp unless she retains full control, thus making it not-quite-decentralized. In Chromia, upgrades can be deployed through a built-in governance and transition mechanism.

Consensus & nodes

Model overview

It is clear that the full node model doesn't scale particularly well. If we require users to run a full node which has a complete copy of the system state then dapps are severely limited in what computations and storage resources they can use.

With the aim of achieving better performance at scale, we propose a model in which individual dapps are hosted on a subset of validator nodes which establish consensus on any modifications to the dapp state, and handle client queries. The system should permit any user to run a full replica node if desired, but the system should not depend on these replica nodes for operations. []

Sybil control mechanism

The research done by our team indicates that commonly used Sybil control mechanisms like PoW and Proof of Stake (PoS) are unsatisfactory[13][14][15]: neither of them guarantees a sufficient level of Sybil attack mitigation, or even a particularly good measure of decentralization. Evidence indicates that most PoW-based blockchains, including Bitcoin, might be de facto controlled by a small group of entities. This problem is particularly bad for smaller cryptocurrencies which do not

[10] A list of the most serious Ethereum vulnerabilities can be found here: https://www.dasp.co/

[11] https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/

[12] https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/

[13] Bentov, I., Gabizon, A., & Mizrahi, A. (January 01, 2016). Cryptocurrencies without proof of work. https://arxiv.org/abs/1406.5694.

[14] Bentov, I., Lee, C., Mizrahi, A., & Rosenfeld, M. (December 08, 2014). Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. https://eprint.iacr.org/2014/452.pdf

[15] https://download.wpsoftware.net/bitcoin/pos.pdf

yet have an independent mining ecosystem. PoS also comes with no decentralization guarantees, and DPoS[16] in particular is prone to formation of cartels and bribery.

Thus instead of following commonly used approaches we will design Chromia consensus and Sybil control mechanisms from first principles.

What Chromia is trying to achieve can be compared to cloud computing: an application which redundantly uses multiple cloud hosting providers can be considered a decentralized application, in the sense that failure or censorship of a single cloud hosting provider does not result in a shutdown of the whole application. A cloud computing model also allows users to use thin clients instead of hosting a complete replica of the application backend on their personal device.

The essential roles in the Chromia model are defined as follows. Chromia software runs on *nodes*, physical or virtual instances of computing power. Nodes are controlled or perhaps owned by some kind of individual, organisation, or collective which we refer to as a *provider*. *Users* connect to such nodes to post transactions, query data or synchronize their private replicas.

A Byzantine fault tolerant network is distinguished from a merely fault tolerant network by its ability to tolerate *arbitrary and potentially malicious behaviour* by network participants. The concept of *nodes* is sufficient for designing a fault tolerant network, but to target proper Byzantine fault tolerance we must account for conscious *provider* entities with the potential to coordinate multiple nodes.

Crucially, to keep a dapp decentralized we need to make sure that the nodes which run its blockchain(s) belong to *different and non-colluding* providers. In that case the application can tolerate a subset of providers experiencing failures, being compromised or performing hostile actions.

For this to work, network participants need to i) know which nodes each provider controls and ii) make sure that providers are *actually* distinct. The latter cannot be done mechanically, but it can be done socially. There is plenty of evidence that Microsoft and Google are different providers, but there's no mechanical way to prove it.

We believe that *all* decentralised consensus ultimately depends on "social consensus". Fully automated decentralised systems are a fantasy, in the end it is people who determine the rules of the system. Chromia acknowledges this, and includes it as a fundamental design principle. In practice, provider distinctness will be achieved as follows:

1. Initially, ChromaWay will select a set of distinct providers. We believe that our extensive knowledge of blockchain and IT industry will allow us to choose well, and we are incentivized to select providers that the users will accept. Users who are concerned about provider uniqueness are welcome to do their own research and contribute to the decision making process.

[16] Delegated Proof of Stake, https://bitshares.org/technology/delegated-proof-of-stake-consensus/

2. Eventually, once the system has a sufficiently diverse set of providers, we will allow providers themselves to vote to add new providers and the system will no longer depend on ChromaWay as a gatekeeper.

Consensus

Each blockchain within Chromia will be associated with a set of validator nodes which is a subset of all nodes belonging to Chromia. This subset of nodes will run a BFT consensus algorithm. Since the set size is limited, PBFT-like algorithms are the optimal choice -- they are well-researched, work well with smaller sets of validators, and provide definitive finality, making reorganization impossible.

However, there are two systemic risks with signature based consensus of this kind which must be considered:

1. The possibility of collusion between providers.
2. The possibility that a majority of nodes might be compromised via a “zero-day” exploit of some kind.

The former risk is extremely subtle, and is discussed at some length elsewhere in this paper. The latter is generally difficult to defend against, the best approach is to encourage a diverse range of software and hardware in the provider ecosystem. Even with mitigation strategies in place, the threat is compounded by the behaviour of signature-based consensus under failure conditions. It has been shown to be prone to catastrophic failure[17], meaning that a breakdown in consensus can corrupt the chain to the point that it becomes very difficult to recover.

For this reason, we decided to implement an additional layer of protection by anchoring blocks in a PoW-based blockchain, such as Bitcoin or Ethereum. This can be done cheaply, a single Bitcoin transaction anchoring the entirety of Chromia every few blocks costs very little, and it will guarantee that Chromia confirmation strength will be *at least as strong as Bitcoin* for blocks which are anchored. For example, a user who prefers to rely on Bitcoin security can wait until an incoming payment is confirmed via Bitcoin anchoring before they send goods.

Node compensation

Dapps require computational resources and storage and should be able to pay providers for them. Providers should be incentivised to offer high quality services to dapps at competitive prices. Chromia will establish a marketplace where dapp developers and node providers can buy and sell resources.

ChromaWay will act as a key node provider in the very early stages. New providers will join as the ecosystem gathers momentum, with lower resource prices stimulating dapps and higher prices stimulating providers. Eventually market equilibrium will be achieved. We estimate that in the long run the cost of using node resources will roughly match the cost of cloud computing platforms like AWS EC2.

[17] https://download.wpsoftware.net/bitcoin/pos.pdf

Miscellaneous features

We believe that to meet the requirements of high performance decentralized applications Chromia has to meet the following requirements:

- Confirmation time: ~1 second (necessary for good UX, real-time user interactivity…)
- Transaction rate: >500 TPS per sidechain. Overall rate in the whole system is unlimited.
- IO capacity: >100k updates and reads per second

Preliminary tests of the Postchain framework demonstrate that it is possible to meet and exceed these requirements.

Chromia will also come with a client SDK which supports development of the client side of decentralized applications. SDKs will be offered for JavaScript (to enable browser-based apps), Java and other languages. The SDK will also enable platform-wide single-sign-on and a wallet for key management to spare users the hassle of registering in each app separately.

Decentralized applications

We assume that the reader of this document is already familiar with the concept of a decentralized application. Nevertheless, it makes sense to clarify to what exactly we refer, as it is connected intimately to the goal of the platform. By 'decentralized application' we mean a multi-user application which is hosted and provided in a decentralized way. That is, no single entity should have control over the functionality of such an application.

Potential problems with central control are that the controlling entity can:

- Shut down the application
- Deny service to certain categories of users
- Monetize users by violating their privacy
- Remove functionality which is valued by users

Open source and peer-to-peer software addressed the centralized control issue for certain categories of apps, such as office software and file sharing, but software which relies on server-hosted databases is much more challenging. Bitcoin was arguably the first to achieve just this, creating a secure decentralized shared database of financial transactions and enabling payment applications outside of the control of centralized entities.

However, bitcoin's "database" is extremely primitive. A more advanced decentralized database makes it possible to decentralize even more applications, and probably to create completely new kinds of applications which were previously inconceivable.

Decentralized applications have the following desirable traits:

- Not controlled by a single entity.

- Ideally, controlled by the community of users.
- Cannot be shut down
- Censorship-resistant -- service cannot be denied
- Transparent, users can see what is going on
- Privacy -- users have control over their data
- Highly available

We do not expect decentralized applications to have *all* of these features. In fact, some features might contradict each other. For example, a dapp may enable the majority of users to restrict access to a minority, in which case the dapp is controlled by users, but it is not censorship-resistant. In practice, application developers aim at a reasonable compromise between decentralization and other priorities.

Transparent apps

Some apps are only partially decentralized: only data which is critical for transparency is hosted in the blockchain, while the rest of the app is centralized. Such applications are better described as transparent apps (tapps) than decentralized apps.

Many apps which are marketed as dapps are in fact tapps. For example, CryptoKitties[18] stores kitty ownership information in the Ethereum blockchain. It can be shut down unilaterally by the company which controls it, and can therefore not be called decentralized in a meaningful sense. It can be shut down in several different ways:

- Shutting down the website. Since the client code is not open source, without the CryptoKitties website it becomes impossible to play the game.
- Shutting down contracts. The company behind CryptoKitties can shut down contracts hosted on the Ethereum blockchain.

Thus, in practice the only thing which differentiates CryptoKitties from a centralized app is transparency.

Token model

Traditional funding and monetization models do not work well for decentralized applications. The value calculation made in a traditional funding model is based on control of centralized ‘property’ like data, user-base, intellectual property, and patents. A decentralized application ideally belongs to its users, a diverse group of stakeholders who form some kind of mutually beneficial balance. There is no central party to own assets, add value, and profit from that activity. That’s why we need a different kind of funding model which is more compatible with distributed ownership. For ownership to be distributed, it is necessary to denote the proportion of ownership or stake in the system with some sort of liquid or semi-liquid asset. This makes it possible to quantify the stake proportion of a given actor, allows them to add value without controlling or submitting to control, and to exchange that value securely. Usually this is achieved with tokens.

[18] A popular game that allows players to purchase, collect, breed and sell various types of virtual cats. https://www.cryptokitties.co/

The basic ICO token model looks roughly like this:

1. Issue tokens.
2. Sell tokens to investors.
3. Do whatever you want with the money.

Instead of that, Chromia will provide mechanisms which balance the interests of developers and users. Essential to this is the Chromia meta-token called Chroma. Dapp tokens can be automatically backed with Chroma, providing liquidity and value which is independent of investment into the dapp in question. Dapp investors can be compensated in Chroma through a profit-sharing contract. For developers, Chromia offers the opportunity to derive income from dapps. This incentivises the creation and maintenance of high quality dapps because better dapps generate more income and create more demand for tokens owned by the developer. The Chromia model is designed to support sustainable circular economies and foster a mutually beneficial relationship between developers, users, and investors.

The role of Chromia

Chromia aims to be the decentralized database component of decentralized applications. A combination of a decentralized database and code, which is run on end-user devices (e.g. mobile or browser app), will typically comprise the entire decentralized application. Let's see how Chromia enables dapp features:

Not controlled by a single entity.

We assume that after creating a dapp, developers would make both front-end and back-end (i.e. parts which run in Chromia) code open source. This allows the app to be used and developed without necessarily involving the original developer.

The data which belongs to the app will be hosted by Chromia. This is done in two tiers:

1. The Chromia root system consists of a diverse pool of nodes which run application blockchains, manage token conversions, allocate node compensation, and other core functionality.
2. Each dapp will select some similarly diverse set of nodes to manage its data.

Both these tiers are decentralized cryptoeconomic systems, and thus we can say that the application is not controlled by a single entity. Typically users will pay for the resources necessary to host the application. A potential problem is that application code might grant control to some centralized entity. Ideally the users should demand an independent review and use the application only if control structures are reasonable.

Controlled by the community of users.

Chromia will include optional governance mechanisms which will allow users to control various aspects of dapp functionality. For example, code upgrades.

Cannot be shut down.

As mentioned above, Chromia enables decentralized application hosting, this ensures that a single entity cannot shut down an application. But we cannot guarantee that an application cannot be shut down by legal action. Chromia root structures will be dominated by few companies (at least within first few years of its existence) which have to comply with laws. Thus an application might have to be evicted from Chromia.

We should note, however, that application fundamentally belongs to users. Chromia is a public hosting platform and completely open source. If users disagree with a government decision to shut down the application, they can simply move their data elsewhere, i.e. they can set up a different Polis (similar to a fork in a traditional blockchain) in a different jurisdiction. As long as users have a need for an application and are willing to support it, it cannot be shut down.

Censorship-resistant.

In the Chromia model, application developers will typically delegate operations to nodes. Nodes process user requests using a consensus mechanism. Thus neither developers nor nodes have the ability to implement censorship on a whim.

It is theoretically possible that multiple nodes can collude to implement censorship, but then users can demand that the application be moved to other nodes. Of course, it is possible that an application would have some censorship components (anti-spam, anti-abuse, etc.) as features. What is reasonable depends on the particular application. If users believe that censorship is unwarranted, they can fork the application and host an updated version.

Transparent.

Application data will be hosted on multiple nodes and blockchain consensus makes it immutable once it’s finalized. We believe that many applications will have transparency as the only feature. Chromia is a neutral technology provider, it doesn’t by itself enforce decentralization. In many cases transparency is already a huge improvement over the status quo.

Privacy.

Privacy is a complex topic. Decentralized application data is typically public, thus the application has to be engineered with that in mind. For example, it might use pseudonymous identities, cryptographic constructs such as hashing, zero-knowledge proofs and so on.

We believe that this approach is better than a traditional approach based on trust and secrecy of application providers. In a centralized model, if a provider’s security is breached, privacy is 100% compromised. In our model, since the data data is public in the first place, it cannot be compromised.

Chromia plans to offer privacy-enhancing features (for use in dapps) in future.

Highly available.

Chromia is designed to withstand node failures. The number of failures it can withstand is a configurable parameter. Minimal number of nodes is four, at that point it can withstand one node failure. If higher availability is desired, a higher number of nodes can be used.

Decentralization quality

Chromia aims to be a neutral technical platform rather than as a moral authority, thus it will allow applications to be hosted regardless of their decentralization level.

However, we believe that decentralization is important, and it's important for users to know features of application they are using. For this reason, we plan to develop guidelines and evaluation criteria. Independent companies will be able to rank applications on these criteria. We also encourage users to demand an independent code audit.

Platform architecture

In this section we describe the platform architecture, expanding on the "Design rationale" section.

Postchain

Chromia is based on the Postchain[19] framework. Postchain defines interfaces between the components of a blockchain-based system and provides a number of building blocks for networking, consensus, cryptography, etc.

The main difference between Postchain and other blockchain frameworks is that Postchain is designed to store blockchain data (both raw blockchain contents and application state) in a relational database. Not only that, Postchain allows transaction logic and consensus to be fully aligned with a relational database; e.g. transactions which violate constraints in the database are rejected and excluded from consensus, they do not result in fatal errors of any kind.

Postchain is implemented largely in Kotlin and runs on the Java Virtual Machine (JVM). The JVM is one of the most commonly used virtual machines, it's geared towards server use cases and has a large number of libraries available. The JVM provides inherent protection against vulnerabilities such as buffer overruns/underruns, data leaks and so on -- it controls access to objects, performs array bounds checking and does not expose error-prone features such as raw pointers. Thus apps implemented on the JVM are usually free of problems such as remote code execution even when they contain bugs. This is very important for blockchain software as remote code execution can lead to huge losses.

Kotlin further tightens type checks and particularly ensures null safety within the code written in Kotlin. We believe that use of a modern programming language designed for safety can reduce the number of defects and help to make sure remaining defects do not lead to drastic consequences.

[19] Source code can be found at https://bitbucket.org/chromawallet/postchain2/

Postchain allows multiple blockchains to be hosted in a single database and allows one blockchain to "see" data belonging to another blockchain when that data is final (committed). This simplifies implementation of inter-blockchain interaction, as blockchains can refer to shared data without any additional overhead or complexity. In particular, this can be used for inter-blockchain asset transfers.

Chains

Chromia is divided in multiple blockchains in order to achieve horizontal scalability. In this model, each node only needs to work with data related to its corresponding blockchains. This architecture increases scalability and simplifies updates, as an update of a single blockchain will have no effect on others.

The overall system consists of a number of "system" blockchains which are essential for Chromia functionality and a number of application blockchains which are specific to particular applications.

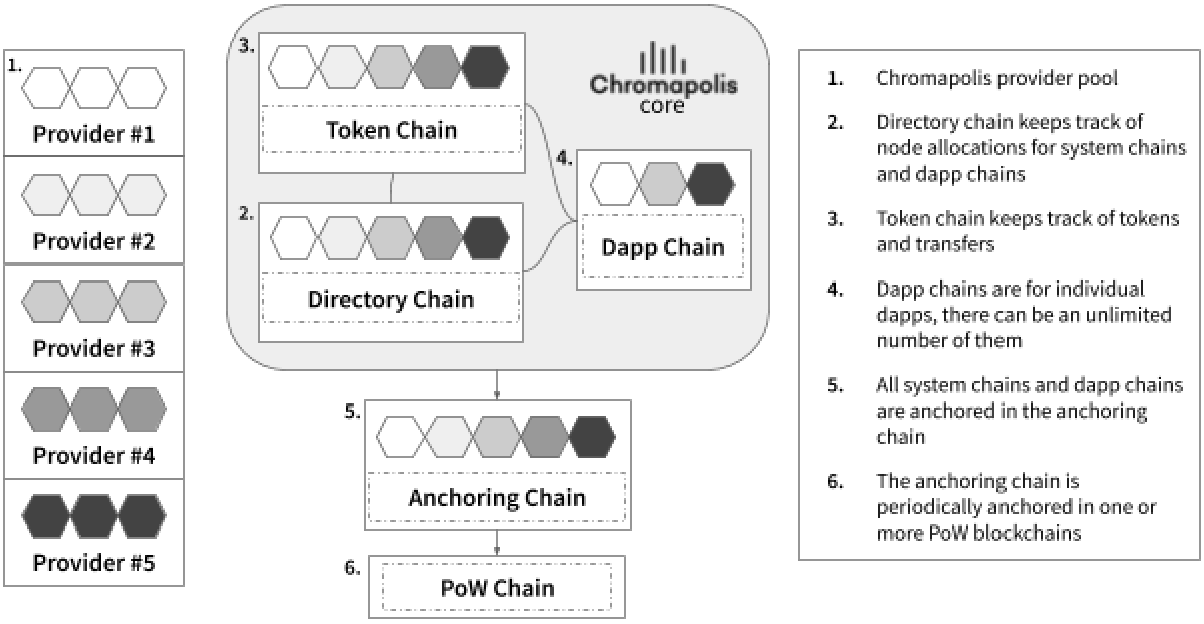
System chains:

Root chain.
Validators: root nodes.
Purpose: keep track of the list of root nodes.
Description: The root chain is needed for thin clients to be able to validate any data within Chromia without downloading the entire blockchain.

Directory chain.
Validators: root nodes.
Purpose: keep track of all providers, nodes, application blockchains and their validators.
Description: The directory chain is responsible for keeping track of all critical information and orchestrating the operations of the system.

Token root chain.
Validators: as defined in directory.
Purpose: keep track of Chroma tokens.
Description: The token root chain keeps track of token distribution between other chains.

Anchoring chain.
Validators: as defined in directory.
Purpose: Defend against attacks on a subset of nodes.
Description: The anchoring chain records hashes of blocks of other chains. This makes it possible to detect consensus failures. In case of a consensus failure, blocks anchored in the anchoring chain take precedence over other versions of blocks. The anchoring chain is itself anchored in Bitcoin & Ethereum blockchains.



1. Chromapolis provider pool
2. Directory chain keeps track of node allocations for system chains and dapp chains
3. Token chain keeps track of tokens and transfers
4. Dapp chains are for individual dapps, there can be an unlimited number of them
5. All system chains and dapp chains are anchored in the anchoring chain
6. The anchoring chain is periodically anchored in one or more PoW blockchains

(Security considerations related to the maintenance of multiple chains are explained in a separate section.)

Node implementation

The data model and operations needed for system functionality such as node selection and rewards can be implemented in Rell. Use of a high-level, declarative language can simplify the implementation and reduce the possibility of defects.

Interaction with other blockchains

Interaction with the Bitcoin and Ethereum blockchains are needed for anchoring. Ethereum interaction is needed to allow ETH to be used for payments within Chromia and for Chroma as an ERC20 token. This functionality can be implemented via indexers: nodes which have to interact with Ethereum need to run an Ethereum node in parallel to Chromia node and import information from the Ethereum blockchain to the Chromia database.

Components

The following is a list of software components which we plan to implement for the Chromia MVP release:

1. Rell compiler and runtime environment
2. Rell IDE: tooling which makes development easy
3. Client SDK: allows front-end (web or mobile app) to connect and interact with Chromia.
4. Chromia node, system chains
5. Bitcoin & Ethereum support needed for anchoring
6. Chroma ERC20 contract, gateway on Chromia side
7. Auto-conversion smart contract on Ethereum

Governance

Chromia supports different governance structures on the system and application levels.

Chromia system governance

System-level governance covers the following topics:

1. System updates, that is, updates to system blockchain structures, their rules and so on.
2. Tuning parameters such as the price of running a dapp according to economic realities.
3. Acceptance of new members into the system.
4. Exclusion of bad actors.

Obviously, governance must be decentralized, a single entity shouldn't have control over the system. We believe that providers are in the best position to perform governance duties:

- They can professionally review the proposals.
- They are motivated to keep Chromia interesting both for users and for application developers. A bad governance decision will affect revenues and profits collected by providers.

Thus we can require ⅔ of providers to vote in favor of a governance proposal to approve it.

Initial centralization

The initial launch of Chromia MVP will likely not have a sufficient quantity of independent providers. Thus at the initial stage governance will be centralized: all decisions will be made by ChromaWay in consultation with system stakeholders. Transition to proper decentralized governance will happen when the system is ready from a technical perspective and the provider ecosystem is healthy.

Rejected alternatives

Stake / coin voting

A widespread governance model in blockchains which do have on-chain governance is stakeholder vote or "coin voting". This is particularly common in DPoS blockchains since stakeholder voting is an essential part of Sybil control & consensus mechanisms. We thoroughly considered this model and rejected it for the following reasons:

1. Usually it is not possible to control stake decentralization, i.e. tokens might be concentrated in a few hands, therefore it cannot guarantee decentralized governance.
2. It's not fair in the sense that rich stakeholders have more power.
3. Many users keep their tokens on exchanges, essentially allowing exchanges to vote for them.

4. DPoS style voting seems to be particularly prone to problems with bribes, cartels and centralization. These problems have been actually observed in the wild.
5. Even if tokens were more-or-less evenly spread, few users actually go through a hassle of voting, few users can understand the proposals, etc. This was demonstrated in the DAO case.

No formal governance

Some cryptocurrencies, e.g. Bitcoin, take pride in having no formal governance. It might work fine if all they want is “digital gold” -- after all, gold itself has no governance. But Chromia is more complex, and it needs to be able to respond to challenges in a timely and coordinated manner, thus Chromia needs a formal governance system.

Unique users

It’s tempting to give one vote to each user -- thus making governance fairer than “voting with money”. But it’s impossible to identify unique users in a decentralized setting, and many issues related to stake voting still apply. Particularly, users might be not informed enough to make good decisions.

Still, we plan to experiment with this kind of governance: our plan is to identify a set of users who actively want to participate in governance -- “citizens of Chromia”. Sybil control can be implemented by keeping track of the social graph. We have no immediate plan to give these users any formal governance power, but they can cast advisory votes.

Application governance

Different applications have different governance needs:

1. Some are designed to be *immutable* and thus would require no governance at all.
2. Other might exercise direct democracy and give each user a right to vote.
3. Another option is to implement weighted voting, e.g. proportional to tokens one has.
4. Dapp developers can also play a role in governance, and either:
    a. Maintain full control
    b. Work together with users through voting, e.g. developer makes proposals which users can approve or reject

We want to give developers and users an ability to decide for themselves and experiment with different forms of governance as they please.However, we want to ensure that users always have certain freedoms:

1. The freedom to access and copy application data. This is an inherent property of a public blockchain.
2. The freedom to fork the application. This is an inherent property of free and open source software and public data: anyone can make a modified copy of software and run it on a copy of data.

Thus we do not impose any restrictions which aren't an inherent property of applications running on public blockchains.

Chromia will provide tooling which would give users the ability to fork a dapp if they are displeased with its governance or just want to experiment with something different. Our goal is to make sure that this forking can be done in a smooth and civil manner.

Uses

Chromia is a general purpose platform suitable for a wide range of applications. But we live in a world with many competing blockchains, thus it makes sense to focus on relative strengths:

- Chromia is database-centric, as such it is particularly suited for applications which are similar to databases in their nature, or deal with complex data schema, complex queries, indexing and so on.
- Chromia has excellent data read-write capacity, thus it's uniquely suitable for applications which require operating on large amounts of data.
- Chromia allows both fast queries and fast confirmations. Thus it is suitable for interactive applications where data needs to be displayed and updated within seconds.
- Chromia is very flexible in terms of resource use policies, thus it can accommodate different business models which do not work on the previous generation of blockchains.

Tokens

Tokens are the bread and butter of blockchains.

- High capacity: we aim to support 50 million token transfers per day per blockchain in MVP version of software. This isn't a world record, but it should be enough to support large user bases. Token transfer capacity can be further improved in future versions.
- Low latency: transfers can be confirmed within 2 seconds which should be enough to support in-person payments.
- Flexibility: token implementation is fully programmable, any imaginable feature can be implemented.
- Custom fee policies: fee policy is decided on a per-dapp basis. This means that transfers can be free, or subject to a flat fee, or a fee proportional to the trade amount.

- Native multi-token support and atomic swapping: trustless token exchange is implemented on transaction format level, it doesn’t even require any special support in the dapp.
- Inter-blockchain transfer: tokens can be moved between different blockchains within Chromia. Non-Chromia blockchains can be supported in future.
- Thin wallet support: a thin wallet (e.g. mobile or browser wallet) can validate transfers within seconds, without syncing with a blockchain.

Games

Blockchain-based gaming is a fast-growing sector of the crypto economy, but current blockchain technology severely limits what games can offer. Typically a blockchain is used only to host tradable tokens, while the actual gameplay happens outside of blockchain.

Chromia can allow much more advanced kinds of games where the entire game world can be hosted within the blockchain, evolving over time according to predefined rules. Updating the game state every cycle requires a number of read and write operations proportional to the number of units in the game. This means that blockchains which do not have high read/write capacity can support relatively few units/players.

On the EVM, loading and storing a memory cell which is already non-empty costs 5200 gas. The block gas limit at the time of writing is 8,000,000. Thus Ethereum can do at most 1500 read/write operations per block. If the entire Ethereum blockchain was dedicated to a single game, at most 6000 units could be updated (e.g. moved) per minute. A Proof-of-authority public blockchain called GoChain offers 136500000 gas per block and 5 second inter-block interval. This implies 5250 cell updates per second.

For Chromia we target at least 100,000 cell updates per second in the MVP release, offering capacity that is twenty times higher than the best available public EVM-based chains. We plan to increase this number in the future with optimized in-memory blockchain state storage.

Here’s a list of Chromia benefits for gaming applications:

- Fast game client load (thanks to advanced query capability the entire game state relevant to the user can be transferred to the client in a matter of seconds)
- Interactivity: updates can be confirmed within seconds, data can be retrieved from blockchain within seconds
- High read & write capacity (upwards of 100k updates per second)
- Good support for complex data schemas needed to support game worlds
- Ability to update code over time

- Comes with game token pegging contracts which can create automatic liquidity for game tokens. Token use in games will be covered in more detail in the “Token” section.

Business uses

Based on our experience with enterprise blockchain applications, we believe Chromia can be used in applications where data is either open, or can be openly hosted in encrypted form, or only commitments (hashes) need to be revealed. This can be particularly relevant in applications which are connected to transparency. Indeed, publishing data via a private blockchain hardly makes things more transparent.

ChromaWay is planning to offer Chromia-based storage option for its Esplix business contract platform, thus allowing businesses to utilize Esplix contracts without a hassle of running their own blockchain nodes.

Tokens and incentives

Similar to how tokens are used in Ethereum to pay transaction fees and compensate block producers, Chroma tokens are used in Chromia to compensate block-producing nodes.
But there is a difference: in the Ethereum model, fees are paid directly by users who make transactions. In Chromia, fees are paid by dapps, which can in their turn collect fees from users. This is discussed in more detail in the next section.

Fees

Application fee models

In Chromia, users pay fees indirectly:

1. The dapp pays to nodes hosting fees. Fee is paid from dapp token account on a daily basis and depends on computational resources requested by the application and used data volume.

2. The dapp itself can collect fees from users according to its own policies.

This means that there’s no system-wide fee policy for users. Dapp developers are free to implement any policy they want. We believe following fee models might be relevant:

1. Classic model: fees are paid for each performed action. Unlike in Bitcoin and Ethereum the price can be fixed, fees do not need to be demand-based.

2. Subscription model: user pays a subscription and then can perform actions without additional payments, however, these actions should be rate-limited to prevent abuse. For example, on a Twitter-like service a user might be restricted to 50 messages per day.

3. Freemium model: certain action might be performed for free, but other actions might require paid subscription. The freemium model is very common for internet businesses.

4. Subsidized model: an application might collect no fees from users, and instead rely on a pre-funded account provided by a sponsor. This can work well when sponsors derive benefit from users outside of blockchain, e.g. the dapp might be available only to users who bought a physical product. This model could work well with manufacturers of IoT devices sponsoring users who bought the device for use of a related dapp.

5. Donation-based model: wealthy donors might donate tokens to provide services to users for free.

6. Gameplay-connected: user can pay fees indirectly when they perform game actions:
    a. Buy in-game items, land, etc
    b. Convert tokens to "game gold"
    c. Trade items
    d. Pay in-game taxes

Hosting fees

In general, Chromia dapp hosting fees depend not on resources *consumed* by an application, but on resources *allocated* for an application. This is similar to how dedicated and "virtual private" server hosting works: the hosting company doesn't care what server is actually doing, it wants to be compensated for providing a server. This is also the model used by AWS EC2, Google Cloud Compute Engine and similar services. In the blockchain space, a similar model is used by EOS.

Applications' needs can be very different. Some applications require a lot of computational resources, some need to process a large number of transactions, some need more storage space, some need a small amount of very fast storage. The kind of hardware which is optimal for an application depends on its requirements.

For this reason, we introduce different node classes. Class requirements will likely evolve over time depending on needs of applications, provider capacity, hardware availability, etc. Provisionally, at MVP launch we want to introduce three classes:

A. The fastest class for applications which require high transaction rate or expensive processing. Specs: 3+ GHz CPU, two hardware threads per blockchain, NVMe storage.

B. Medium class. Specs: 2+ GHz CPU, 1.5 hardware threads per blockchain, SSD storage.

C. Economy class. Specs: 1+ GHz CPU, equivalent of a single 1 GHz hardware thread per blockchain, SSD storage.

Application hosting fee paid on daily basis is split into several components:

1. Percentage of processing time.

2. Number of transactions.

3. Storage.

Chromia doesn't have the means to precisely measure computational resources "consumed" by an application as this depends on a variety of complex factors which are outside of control of Chromia code (CPU caches, CPU pipelining, OS context switch overhead, DB engine optimizations etc.). Instead Chromia will measure the median time taken to process a block as reported by block producer nodes.

When a thread allocated for an application never goes idle (i.e. it continuously builds or applies blocks), the application is using 100% of processing time. In that case it pays a full price for one day of hosting for a particular class.

When an application uses less than 100% of processing time, it's eligible for a discount. For class A and B nodes, the discount is limited to 50%. Even if the application is completely idle it still has to pay half of the day hosting price. This is necessary because actual physical resources are allocated to an application whether it uses them or not. A limited discount is provided because we want to encourage applications to be as efficient as possible. Idle time might increase capacity available to other applications, decrease energy consumption and hardware wear.

For class C node hosting there is no limit to discount and applications which build no blocks will pay nothing in hosting costs. Additionally, class C allows applications to specify throttling. An application that doesn't want to pay more than 50% of the daily hosting fee can be throttled to use no more than 50% of processing time. Class C nodes will use special algorithms which allow efficient co-hosting of a large number of blockchains. As a result of this, class C nodes target rather than guarantee their posted capacity.

Storage costs and per-transaction costs also depend on the class of nodes used by an application. Hosting 1GB of data on class C nodes will be much cheaper than hosting the same amount of data on class A nodes.

The hosting price is standardised by selecting the median of prices submitted by all providers. A more sophisticated market which allows providers to auction spare capacity will be developed in the future once the number of providers exceed decentralization needs.

Node incentives

The block building process should be properly incentivized. That is to say, it should not be profitable for nodes to neglect their duties e.g. by making only empty blocks or no blocks at all.

In theory the collective of providers has an interest in offering a great service to all applications. If applications move to other blockchain platforms, providers cease to make any money. However, we also need to consider providers who might try to cheat the system for individual gain. Beyond the basic incentive to not create invalid blocks or conflicting histories (which can be automatically detected and punished by automatically excluding a node, and possibly its provider, from the system), the system can track the following data:

1. Number of blocks built by a node for a particular blockchain as a primary (the role of primary is rotated over time).
2. Number of transactions in blocks built by a node as a primary.

3. Number of commit messages submitted.

This data can be used to detect nodes which neglect their duty as a primary or are not fast enough to submit commit signatures. Nodes which systematically underperform can be excluded automatically or through a providers' vote.

Note that nodes of as a whole have an interest in accepting as many transactions as possible and storing as much data as possible as they are paid by number of transactions and storage used.

Another resource which other blockchain systems typically neglect is a node's capability to reply to queries. Indeed, if nodes are compensated only for the amount of data processed, they are incentivized to ignore queries and only process transactions. But if users run light clients, queries are absolutely crucial. We have developed a mechanism which creates an incentive for nodes to reply to queries. It is explained in detail in the Appendix. Simply put, upon receiving a response from a node a client can discover that this response is "lucky" via a mechanism similar to PoW. Only a fraction of all responses (e.g. 1 in a million) is "lucky". A lucky response is published in a certain blockchain and yields a small reward both to user and to the node which produced the response. Special provisions (covered in the Appendix) are made to discourage nodes from farming lucky responses on their own.

Node stakes

To encourage providers to secure their nodes they will be required to put Chroma tokens into a separate account which represents the provider's stake in the Chromia economy and is used as collateral which is forfeited when nodes owned by a provider misbehave.

Providers can group nodes into units with different levels of stake: high, medium, low. High-stake nodes should be more thoroughly secured as they can be used for applications highly sensitive to security, such as running system blockchains and high-volume financial dapps. Low-stake nodes can be used for less sensitive dapps such as simple games. Each dapp can specify a minimal stake which is required for nodes which run it. The stake level necessary for system blockchains is set by a council of providers.

Token use in games

The current generation of blockchain games are based on collectible items and do not offer rich gameplay. We envision a new generation of massive multiplayer online games with rich game worlds hosted within Chromia blockchains, and rich market economies based on tokens and tradeable game items.

For this kind of game Chromia can offer a set of smart contracts which make game tokens liquid and valuable. This would allow game developers to quickly bootstrap game economies. For game users, pre-made smart contracts offer a certain degree of stability: they can be sure that game tokens they earn won't lose all of their value overnight due to a poorly coded token structure.

At the heart of Chromia game smart contracts, is a market making/token conversion algorithm similar to a widely known "Bancor algorithm" (a similar algorithm was discovered by Chromia team members before Bancor). When Chroma tokens are converted to game tokens (e.g game “gold” tokens), new game tokens are created. Chroma tokens are put into the smart contract reserves and the price is adjusted. Price adjustments work in such a way that higher demand (more people buying game tokens than selling) results in a higher price. When game tokens are converted back to Chroma, the price is reduced. The algorithm can be configured to enable smooth price movement, so the game token price against Chroma cannot drop significantly unless the vast majority of users abandon the game and convert their tokens to Chroma.

A fee can be collected upon conversion by adjusting the buy/sell price. For example, a 1% fee can be taken out of the Chroma amount and used to:

- Pay for game dapp hosting fees (i.e. it’s transferred to dapp’s hosting account)
- Pay the game developer and, possibly, investors

Game token price increasing with demand means that players have an incentive to invest into game gold. In fact, they have an incentive to discover new interesting games which are going to grow in popularity over time. Indirectly they also have incentive to promote and share the games they play. This set of incentives can result in healthy game community dynamics.

The full list of Chromia features developed specifically for use in game applications will be published in a separate paper.

Chroma token economics

To summarize, the Chroma token has the following roles in Chromia:

- It is used by dapps to pay hosting fees, thus compensating the nodes.
- It is used as a “standard” currency within the Chromia economy, as dapps can collect it as fees, or use as reserves to peg their own tokens, etc.
- It is used to make sure that providers have a stake in Chromia ecosystem thus offsetting incentives to collude.

Since Chroma tokens are used for stake and reserve purposes we expect a significant amount to be taken out of circulation and “locked” for this kind of use.

System accounts

Chromia has several special Chroma token accounts which are used for system-wide purposes:

- ERC20 token pegging: Chroma tokens on this account belong to owners of Chroma ERC20 tokens which enable some interoperability with the Ethereum blockchain. This account is managed by the Ethereum gateway blockchain.

- System node compensation pool: Nodes which run dapp blockchains are compensated by dapps. But nodes running system blockchains also need to earn money. For this reason a certain percentage (decided by the council of providers) is taken out of hosting fees and sent to the system node compensation pool, which is then used to compensate nodes for hosting system blockchains. In other words, Chromia itself can be seen as a dapp which orchestrates and taxes other dapps.

- Future development pool: Initially ChromaWay and its subsidiaries will develop Chromia, but eventually this role should be decentralized. Once the economy is sufficiently decentralized the "future development pool" can be unlocked and used according to providers' vote to improve Chromia as a whole.

- Charity pool: in certain situations where tokens are sacrificed (explained below) a fraction of such tokens can be diverted into a charity pool. Funds from this account can be used to donate to charities according to users' votes. This can promote Chromia as an ethical and socially-conscious blockchain.

Public good account

In certain situations tokens need to be "sacrificed" (irreparably destroyed or burned) to avoid a conflict of interest or the possibility of abuse. These situations include Sybil control mechanisms, punishment of misbehaving parties or a "neutral action" in a case of a disagreement between two or more parties.

Chromia offers an alternative to irreparable destruction in the form of a public good account. This is a virtual account which automatically distributes received tokens into 4 different accounts:

- 25% of tokens are burned; burning tokens indirectly benefits all Chroma token holders as tokens are permanently removed from circulation
- 25% of tokens are put into the "System node compensation pool"
- 25% of tokens are put into the "Future development pool"
- 25% of tokens are put into the "Charity pool"

Thus all Chromia users indirectly benefit from the public good account in the long term. It's very unlikely for the public good account to be abused as control funds in it are collectively controlled and aren't easily accessible. It is therefore a viable and productive alternative to simple "burning".

Funds will be sent to the public good account in following cases:

- A user would need to send 10 Chroma tokens to the public good account to become a "Chromia citizen". This confirms the user's commitment to Chromia and gives certain perks such as the ability to vote, priority services, the ability to participate in "lucky request" reward program, etc. (Details about this program are covered in the Appendix.)
- The lost stake of misbehaving nodes is sent to the public good account.
- 0.1% of application hosting fees are sent to the public good account

Additionally, we encourage dapps to use the public good account when the destination of tokens is unclear for some reason, or if tokens need to be destroyed for game-theoretic reasons. For example, “Burnable Payments” is a simple game-theoretic mechanism which makes sure that that neither buyer nor seller have an incentive to cheat: if the buyer disagrees with the seller he can burn escrowed funds.

Token distribution

One billion tokens will be created upon launch of the system. That constitutes the token supply limit, no tokens will be created in the future. Initial distribution of tokens:

- 70% owned by ChromaWay through its subsidiary Chromia Devcenter Oü to be sold, awarded to team members, invested or used in any other way
- 3% put into an automatic conversion contract on Ethereum blockchain to enable Chroma<->ETH conversion
- 2% put into system node compensation pool
- 25% is allocated for promotional use: to be given to users to try applications hosted on Chromia

Within the ChromaWay allocation, up to 25% (of all tokens) will be sold initially to select partners. The rest will be locked and released slowly over time. Up to 17% will be unlocked during the first year after launch, then up to 12% per year. ChromaWay and its subsidiaries will hold tokens for at least three years. This creates long-term incentive for Chromia development. After three years Chromia development and governance must transition to a decentralized model.

Promotional tokens will also be initially locked, and unlocked at a rate of 0.5% per month.

Thus percentage of tokens in circulation changes over time:

1. At start: up to 30%
2. After 1 year: up to 53%
3. After 2 years: up to 71%
4. After 3 years: up to 89%
5. After 4.5 years: 100%

Promotional token fund

The use of the promotional token fund will be initially controlled by Chromia Devcenter. Its purpose is to encourage use of Chromia platform and dapps hosted on Chromia. Tokens from this fund should be given only to end users, they shouldn’t be used to fund development of projects.

The rationale for this fund is that it’s hard for an average internet user to acquire tokens: they need to register on crypto exchanges, which is a lot of hassle. Also people are generally reluctant to spend money just to try out a new app (which might be not-so-great).

Thus it is necessary to give away tokens for free to build a mainstream user base.

However, this needs to be done with caution. Obviously, Sybil control measures should be used -- it is certainly possible that somebody will try to impersonate multiple users to acquire a large number of tokens for free. One possible way to mitigate abuse is to require some form of identification (e.g. Facebook account).

Tokens might also be given out for use within a specific app or game.

Tokens from promotional fund are released gradually to:

- Onboard users as the system grows
- Monitor the situation and experiment with different ways to distribute tokens
- Avoid disrupting the value of Chroma

1% per month is a *maximum* distribution rate. If promotional use is deemed inefficient tokens might be reserved for later use or sent to the public good account.

Decentralization

Centralization necessary at start

Chromia shall be a true decentralized platform for decentralized applications: not controlled by anyone, open for permissionless innovation.

Decentralization is not a starting point, but a goal. Proper decentralization requires a strong community with a large number of independent participants who are committed to Chromia. But building a community takes time. The platform needs to prove itself before it's deemed interesting enough to contribute to it.

Thus Chromia will be centralized at start; we believe it's better to embrace this and use a centralized development and governance model to speed up development than to play pretend-decentralization.

For this reason, ChromaWay opened a for-profit company called Chromia Devcenter Oü which will act as Chromia development center at the initial stages. As the largest holder of Chroma tokens which are locked for 3+ years, Chromia Devcenter is motivated to increase the value of Chromia as a system, as that will likely also increase the value of its holdings.

After observing the cryptocurrency ecosystem for 7 years, we believe that a for-profit model is the optimal way to scale the development in the initial stages. Here are some examples of failures of more decentralized community-based models:

- The colored coins project suffered from slow development and fragmentation. Even monetary bounties didn't help to attract a persistent developer base[20]. Developers who

[20] See interview with ChromaWay CTO, at the time leading the colored-coins project, on Coindesk in 2013| https://www.coindesk.com/colored-coins-paint-sophisticated-future-for-bitcoin/

joined the project temporarily produced low-quality code and then hopped to something else.
- The Mastercoin project (now known as Omni) bounty-driven process produced three incompatible implementations. Eventually they switched to a centralized process and achieved better results.
- Ethereum Foundation failed to create a working light wallet for three years. As a result, users had to rely on unsafe web wallets, or struggle with keeping their full node wallet in sync.

As a for-profit company, Chromia Devcenter will be able to set concrete goals and focus on them; particularly, focus on features which are essential for Chromia platform adoption and user base growth.

Beyond development, Chromia Devcenter can also

- Organize promotional events
- Help companies to build dapps on Chromia
- Collaborate on projects with other companies
- Invest into the dapp ecosystem

We believe that these activities are better done on a for-profit, commercial basis. Non-profit foundation models can result in inefficient use of funds, abuse, corruption, etc.

It's important to highlight that Chromia Devcenter is **not** Chromia. Once launched, Chromia as a network will have a certain degree of autonomy. Chromia Devcenter cannot force people to run a particular version of software. It also cannot modify any blockchain records or state beyond what it was explicitly granted access to. Thus it cannot be held responsible for what happens within the network.

With respect to Chromia network, the role of Chromia Devcenter is the following:

- Produce free and open source node software, which can be independently inspected and modified as needed.
- Control certain parameters such as the pricing of resources and selection of providers until the network is big and decentralized enough to control these parameters on its own.

There are two risks associated with this role:

- Node software (or other relevant software) will have a backdoor or other security threat. *Mitigation*: We encourage providers and users to review software before running it.
- System parameters or provider selection can be set to values which disrupts the system. *Mitigation*: We will limit the rate of change via blockchain rules enforced by nodes. In the worst case providers/users can fork the network to avoid disruptive settings.

Decentralization through a diverse set of providers

Once the provider ecosystem is mature enough, governance can transition to a group of providers. How does this compare to the quality of decentralization seen in other blockchains?

Bitcoin

Satoshi originally described Bitcoin as “1 CPU = 1 vote” kind of a system. The original user base mostly consisted of ordinary internet users interested in P2P systems, and block production was extremely decentralized. Still, Satoshi was essentially the dictator and could change code as he wants. He could, in principle, make an update which would steal coins from other users.

Eventually the situation with code updates became better: all code which goes into Bitcoin node software is reviewed, Bitcoin node binaries are built using a Gitian process which allows multiple parties to verify that code in the repository corresponds to binaries, this means that end users can rely on a decentralized group of developers to control for possible backdoors and other issues.

On the other hand, the situation with block production became worse over time. First, users joined “mining pools” to make rewards more predictable. As a result they no longer produce blocks, but instead rent their hashpower to a pool which produces blocks. This means that a mining pool can, in principle, produce a malicious chain of blocks. In theory, users should notice this and switch to a different pool, but it will take some time. At a certain point of time, a single pool (GHASH.io) had >50% of total hashpower, and users did nothing.

Another problem came with the advent of ASIC mining: ASIC manufacturing companies started mining on their own. Companies which had more efficient chips got higher profits and could reinvest it into expansion. Economies of scale create a positive feedback loop where production of mining chips and mining itself becomes more and more centralized.

This culminated in Bitmain shipping more than 70% of all mining equipment, and Bitmain-affiliated mining pools having more than 50% of total hashrate. Bitmain doesn’t report any statistics, but we have every reason to believe that warehouses with the Bitmain logo on them full of Bitmain miners actually belong to Bitmain and are the source of an enormous hashrate. In any case, the currently biggest three mining pools can control the network, and two of them are affiliated with Bitmain.

It is also undeniable that the majority of hashpower is hosted inside China, thanks to cheap power, cheap facilities and so on. This gives the Chinese government the possibility to control Bitcoin. It could potentially seize facilities and execute a 51% attack, or a soft fork to introduce censorship.

PoW centralization resulted in delayed network updates and Bitcoin become practically unusable for payments due to extremely high fees. Summary: while Bitcoin development is decentralized, block production is heavily centralized.

DPoS

It was observed that DPoS-based blockchains -- BitShares, Lisk, ARK, STEEM, EOS -- have a large degree of stake centralization, which means that few large token holders can effectively control the network. Problems with DPoS centralization are thoroughly explained by Vitalik Buterin[21].

Ethereum

Ethereum block production is currently PoW-based and thus has roughly same problems as Bitcoin (the three biggest pools can control block production).

It is meant to eventually to transition to proof-of-stake. That doesn't mean that every stakeholder has a chance to produce a block. Instead, the number of block producers will be restricted to about 1000 entities, thus smaller token owners have to delegate block production to pools in order to participate.

Chromia

It appears that no existing projects give control of the network to a very large set of people. Neither does this appear to be a particularly useful approach, most people do not have enough technical knowledge and motivation to keep the network safe. A person who runs software without carefully inspecting it is essentially just a proxy for the entity which decided what software to release.

For this reason we believe that the Chromia model where the network is controlled by a limited group of providers is not an impediment to decentralization. As long as these providers are truly independent, pursue their own goals (i.e. profit from hosting dapps), and operate in many different countries, the system can be considered decentralized.

Initially we plan to get at least twelve providers. In the long run the number can reach thousands, on par with the PoS scheme proposed for Ethereum.

Another way to look at it is a barrier to entry. A Bitcoin ASIC miner can be purchased for several thousand dollars, but a user won't be able to generate any blocks on his own. To become a significant player one needs hundreds of millions of dollars in capital to acquire hardware and build facilities.

On the other hand, any professional hosting company can become a Chromia provider and participate in building blocks. Thus we believe that the barrier for entry is actually lower than that seen in other blockchains.

Number of full nodes

Public blockchains such as Bitcoin and Ethereum boast a large number of full nodes -- estimated to be in the 5000-10000 range. While a large number of full nodes potentially increases network resilience, it also has downsides: the network can't be faster than its slowest node. Thus both

[21] https://vitalik.ca/general/2018/03/28/plutocracy.html

Bitcoin and Ethereum severely limit number of transactions, as well as computational resources required to process transactions.

Chromia takes a different trade-off: number of full nodes might be limited to number of block producers, which will be typically on the scale of 10-100 nodes per blockchain. Does this result in lower network resilience? Let's analyze different threats:

1. **Node hardware failures**: Assuming failures are random, it's extremely unlikely that 10 nodes will fail at the same time, before new replicas can be made.
2. **Network DoS**: While in certain scenarios a bigger number of nodes is helpful, a network can be effectively disabled by specifically targeting block producers, and the number of independent node producers might be actually higher in case of Chromia.
3. **Network partitions**: Networks based on PoW consensus typically do nothing to detect network partitions, thus they can simply work through minor disruptions. But in case of a major disruption it might result in double-spends on the different sides of the partition. The fact that a Chromia-style network stops in case of a partition is actually a feature, not a bug.

It should be noted that Chromia does not discourage users from running full nodes. Every blockchain running on Chromia should be public, therefore any user who wishes to run a full observing node for a particular blockchain should be able to do so, as long as he has access to modern hardware.

In fact, if we compare Chromia to Ethereum, it can be said that Chromia architecture makes it easier to run a full node: an Ethereum user is forced to download data of all dapps and all users. Chromia user can choose what dapps he is interested in and sync only the corresponding blockchain data.

The number of Chromia full nodes might be lower not because it's harder to run a node, but because with a properly working light client it's just not necessary, and we expect that fewer hobbyists care about specific dapps than about "the world computer".

Security

Blockchain

The role of the blockchain is to make sure that there is a single application state seen by all users, and that double-spend and replay attacks are not possible.

In a light client security model, blockchain nodes also take responsibility for validating state transitions and transactions. We will discuss light client security in a separate section; in this section we will focus solely on the security aspects of the "full node" model.

The most basic threat we are protecting against is a single node wilfully violating the rules of the system. This can occur because whoever controls it has become corrupt for some reason[22], or because it has been compromised by an external attacker. Centralized systems built using a traditional software architecture have no protection against that -- a single compromised server can result in arbitrary data modification, which in case of financial data can lead to arbitrary losses. Particularly this might happen in the following scenarios:

- External intrusion through exploitation of a software or hardware vulnerability.
- Rogue employee -- system administrator or other person who has access to the server can exploit it for personal gain.
- Hosting provider's tampering -- physical access to server allows provider to modify data.
- The company itself can arbitrarily change data or rules to its own gain.

The first layer of protection against these scenarios is application logic which requires both cryptographic authorization and a deterministic computation model. When a user's nodes have complete data they can detect cases where rules are violated and thus reject a false application state. In Chromia this is accomplished by requiring applications to be developed in Rell: Rell has a deterministic computation model and makes it easy to implement cryptographic authorization for all data mutations. The overall architecture also makes it possible for user nodes to receive full input data (blocks and transactions) and independently compute the application state.

A more sophisticated attacker can exploit situations where multiple valid application states can exist at the same time. This attack is usually described as double-spending, for example:

1. An attacker produces an application state in which a merchant was paid to ship some goods.
2. The merchant ships the goods.
3. The attacker replaces the application state with another where the merchant is not paid, instead the funds are directed back to the attacker's account.
4. Now the attacker has both the goods and the money.

Many variations of this attack exist. For example, it can be done using different kinds of tokens and merchant can be an exchange.To protect against this attack the system mutually incompatible application states aren't allowed to exist.

This can be done using a Byzantine Fault Tolerant (BFT) consensus algorithm which "confirms" a single application state and rejects all incompatible states after that.

It has been demonstrated that in an asynchronous network (i.e. without confirmation of packet delivery) a BFT consensus algorithm can tolerate up to 33% of node failures. Strictly speaking, ⅔

[22] Corruption here encompasses a range of possible scenarios in which the node provider has incentives to act in a way which is detrimental to the goals of the collective in which it participates. Financial gain, coercion, deception, mental instability; there are many reasons why a node operator might become corrupt.

plus one node must remain honest. For example, a system with 10 nodes can tolerate up to 3 failures, i.e. it will keep working when 3 nodes are compromised.

Chromia uses a PBFT-style consensus algorithm to build the blockchain. When the number of blockchain's validator nodes is 3f+1, a block must receive 2f+1 "votes" to be confirmed (i.e. more than ⅔ of all votes). Users' nodes only deal with blocks which are confirmed.

Thus Chromia can tolerate arbitrary corruption of a minority (less than ⅓) of blockchain's validators nodes with no drastic consequences except a possible slowdown. Chromia will attempt to ensure that any blockchain will be allocated nodes from different providers so that a single failure cannot result in blockchain corruption. Requirements will be especially stringent for system chains.

This is the fundamental assumption of Chromia -- individual nodes (as well as individual providers) can and will fail, but it should have no effect on Chromia users.

But we also need to consider situations when more than 33% of validators for a particular blockchain fail. We consider this unlikely, but possible. While we cannot guarantee smooth operation in case of a "34% attack", we can try to minimize the damage and enable speedy recovery . Particularly, Chromia needs features to:

- Make it difficult for attackers to profit from the attack.
- Make it possible for the Chromia system to detect the attack as early as possible, so that recovery steps can be taken.
- Make it possible for Chromia users to detect the attack as early as possible, so they can abstain from operations which might result in financial loss.
- Allow Chromia users to wait for stronger confirmations for high-value transactions if desired.

The most powerful tool at our disposal is anchoring -- a way of boosting the confirmation strength of one blockchain using another. Let's consider the simplest anchoring scheme. Suppose we want to anchor blocks of blockchain X in blockchain Y. To do that:

1. When block **X_i** is confirmed in blockchain X, one of the block producers will publish a tuple **(X, i , hash(X_i))** in blockchain Y
2. Once that publication is confirmed in blockchain Y, a user's node (which follows both blockchain X and blockchain Y) can find the first tuple of form **(X, i, *)** which is published in blockchain Y
3. Block X_i is said to be anchored when **(X, i , hash(X_i))** is the first such tuple.
4. If consensus on blockchain X fails and a different block **X'_i** is produced, the anchored block **X_i** should take precedence. That is, in case of a recovery the blockchain should include the last anchored block, and blocks incompatible with it must be deleted.

It's easy to see how merchant can use anchoring to boost confirmation strength. Suppose merchant will wait until block **X_i** which contains a payment to him is both confirmed and anchored before he ships the goods. In In this case if consensus of blockchain X fails (e.g. X's nodes are compromised and produce several incompatible histories), but blockchain Y stays correct, merchant does not suffer a loss -- once blockchain X is restarted (e.g. with new validators) the block X_i will be included and thus merchant will receive the money.

Technical implementations of anchoring might differ in:

1. What is being published (e.g. just a commitment)
2. Who can publish information
3. Whether light-client proofs are possible
4. How easy it is for a node to detect anchoring failure

Chromia will make use of multi-level anchoring, that is, blocks from a dapp blockchain will be anchored in a special anchoring-chain maintained by another set of nodes.

Let's consider an example. First we consider the situation without anchoring. Suppose dapp blockchain A is run by 10 validator nodes all of which are compromised. If tokens from this dappchain are traded on a centralized exchange, compromised nodes might be used to perform an attack:

1. Nodes will prepare two versions of a block at the same height: block **X_i** contains a payment from the attacker to the exchange, and block **X'_i** doesn't
2. The exchange sees block **X_i** and credits tokens to the attacker's account.
3. The attacker sells his tokens for bitcoins and withdraws bitcoins from the exchange.
4. Block **X'_i** is revealed to all other nodes and subsequent blocks are built on top of it.
5. It's not possible to tell whether block **X_i** or block **X'_i** should take precedence. Obviously, block **X_i** is better for the exchange, but block **X'_i** might include other important payments.
6. Thus the exchange might suffer a loss even after faulty nodes are replaced, as the blockchain might be built on block **X'_i.**

In a situation with anchoring, the exchange can protect itself from this risk. It should wait until the payment is anchored in block **X_i** before crediting the money. In this case even if nodes try to build an alternate block **X'_i**, that block won't be included into a blockchain after nodes are replaced as it is not anchored.

The merchant can suffer a loss only when the anchoring chain itself is compromised. However, the Chromia anchoring chain will include a larger number of validator nodes from different providers, say, a hundred. It might be enough to compromise 4 nodes to compromise the dapp blockchain, however, it will take at least 34 nodes/providers to be compromised to anchor two incompatible blocks. That is, it requires a collusion on a large scale.

However, we cannot completely rule this situation out. For this reason, the anchoring chain will itself be anchored in PoW blockchains -- Bitcoin and Ethereum. An exchange wallet in high-security mode can wait until block **X_i** is anchored in block **A_j**, and block **A_j** is itself anchored in Bitcoin's block **B_k.** In this case to revert a payment one would need to compromise a significant number of

Chromia nodes, and, on top of that, perform a Bitcoin blockchain reorganization. We believe that this situation is exceptionally unlikely.

Confirmation strength can be further boosted by anchoring to multiple blockchains. Particularly, we consider establishing a network of notaries and highly reputable institutions in multiple countries. If we anchor the Chromia anchoring chain in this notary chain it will be impossible to revert Chromia blocks without a worldwide conspiracy.

Node security

We believe that a collusion among Chromia providers is unlikely as loss of stake, profits and, possible legal action serves as a deterrent. However, if a software exploit which allows an attacker to execute arbitrary code on a Chromia node were to be discovered, multiple Chromia nodes could be compromised at the same time.

The most common causes of remotely exploitable vulnerabilities are memory corruption bugs within the application code. For this reason, Chromia is implemented using a safe language which protects against memory corruption (Kotlin) and is executed on the JVM which provides memory safety.

Another possible source of vulnerabilities is dapp code. Chromia dapps will be implemented in Rell which is itself a memory-safe language; on top of that the Rell execution environment is implemented in Kotlin and runs within the JVM, thus for an application to break out of the sandbox it will need to defeat both Rell and JVM safety mechanisms, which we believe is practically impossible.

The remaining source of vulnerability is code written in C, that is the host OS (e.g. Linux) and DBMS (e.g. PostgreSQL). Exploitable vulnerabilities in the Linux kernel itself seem to be extremely rare, and access to PostgreSQL will be mediated by Rell which limits the possibility of attacks.

Nevertheless, we will research further options to reduce the possible attack surface:

- Run security-oriented Linux distribution with all non-essential components disabled
- Consider using an OS which further reduces the footprint or attack surface (e.g. OSv[23])
- Consider switching to a JVM-based database engine, or implement a new database engine specifically for Chromia

Another possible attack vector is hardware and firmware. For example, the Intel Management Engine is present in the vast majority of Intel's products, and effectively runs a separate OS which can potentially be compromised. This could provide a vector to compromise the node running on the same CPU. To mitigate this attack vector we will recommend providers to diversify and use hardware from different vendors. We will also advise providers to limit their exposure to cloud providers. If the bulk of Chromia nodes run on, for example, AWS, Amazon has a power to fork or shut down the network.

[23] http://osv.io/

Governance security

Governance can be a source of security problems. We can take an example from the corporate world: while the CEO herself might not be able to tamper with servers directly, she can replace the system administrator with one who will, for example, delete some crucial data.

Chromia governance mechanisms must therefore also be designed with security in mind. Particularly:

1. It should not be possible to introduce changes which can be used to fork or destroy blockchains.
2. All changes must be applied with a delay so that they can be reviewed and, if necessary, mitigating measures can be taken, in the most severe cases this might be an emergency hard fork.
3. The rate of changes should be limited.

Light client security

Most Chromia users will use light clients which do not process the entirety of blockchain data.
They will have to rely on Chromia nodes to query data from the blockchain state, and supply the confirmation status of transactions and payments. How can light client users authenticate this data? In short, they have to trust validator nodes. Each block is signed by a BFT majority[24] of all validator nodes. Thus, to confirm an invalid transaction, more than two thirds of validator nodes would need to be compromised.

Light client security isn't significantly worse than full node security. Just over one third of nodes need to be compromised to produce a fork, but more than two thirds need to be compromised to produce an invalid block. The first scenario is far more likely, and light clients are protected against it to the same extent as full nodes.

Light clients can also take advantage of anchoring, including anchoring into PoW blockchains. Anchoring methods used in Chromia can produce compact proofs, this means that a user can benefit from anchoring without needing to run a full node.

In some cases the data retrieved from nodes is not important and does not need to be authenticated. In scenarios where the data does need to be authenticated, different data structures can be used depending on the nature of the data:

1. Transaction Merkle tree: can be used to check that a transaction is confirmed and is valid. E.g. this can be used to verify a payment. The transaction Merkle tree root is present in the block header and is signed by nodes as a part of the consensus algorithm.
2. State commitment Merkle tree: Typically a block header will commit to the set of rows which represent the blockchain state. This allows a light client to make sure that a certain row returned in response to a query is present in the latest blockchain state. State

[24] ⅔+1 of total validator nodes

commitments can be disabled in high-performance blockchains as they increase blockchain overhead. A state commitment Merkle root is present in the block header and thus signed in same way as the Transaction Merkle root.

3. Assertions and indexers: Special data structures can be used to prove that the entire query response is correct and doesn't omit any data. If present, they are signed in the same way in the block header.
4. Signed query responses: When a query response is important but cannot be proven through indexers, a light client can submit requests to multiple nodes and receive signed responses.

A light client can authenticate data via validator node signatures only when it knows validator node public keys. Validator node pubkeys can be obtained from the directory chain. The directory chain itself can be validated using the root chain. The resolution process is as follows:

1. A light client comes with a built-in, hard-coded hash of the genesis block of the root blockchain as well as an initial list of root node public keys.
2. A light client downloads the entire root blockchain to get an up-to-date list of root nodes. Root chain is extremely sparse with only one block per day, and thus this operation is not a lot of burden even to a light client.
3. A light client can query any directory chain replica to retrieve a list of validators for the blockchain it is interested in and validate them with a state commitment mechanism, i.e. checking the signatures of root nodes.
4. Results of query to the directory chain should also be confirmed via the anchoring chain and PoW anchoring.[25]

Dapp client and wallet security

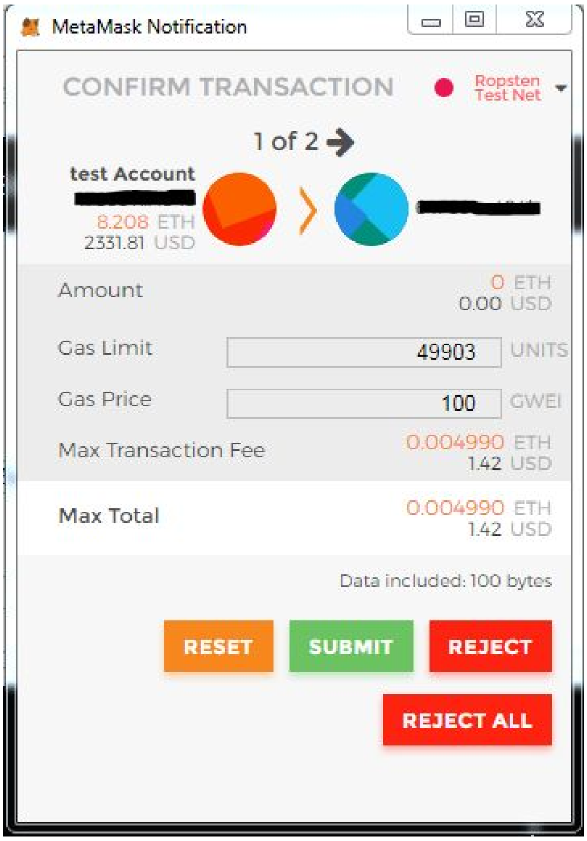
The Chromia team will develop ChromaWallet -- a wallet which can be used to hold Chroma tokens as well as any tokens on any Chromia blockchain which follows the FlexibleTokens standard. It will also provide an ability to interact with dapps using a simple form-based interface and to manage dapp accounts. ChromaWallet will be provided in desktop, mobile, and web app formats and will target hardware wallet integration.

In a future version ChromaWallet will be able to function as a general-purpose dapp browser, sandboxing dapp UI code execution and offering graphical interface rendering on a web technology stack. The dapp browser will be able to download dapp UI code from a Chromia blockchain. Of course, it won't be able to guarantee that this code is free of bugs or security defects, but it will be able to ensure that code can only be updated together with the dapp itself and that all users run identical code (i.e. code cannot be bugged specifically for one user). Note that the dapp browser functionality won't be present in the MVP version.

Instead, dapps which require complex UI, such as games, can be implemented using a separate client delivered as a web or mobile application. In this case security can be controlled through the use of sub-accounts. The dapp client will receive a private key of a sub-account which belongs to a

[25] Step #4 is necessary to makes sure that a collusion of root nodes cannot compromise any other blockchain.

user and will be able to sign transaction on behalf of the user. This means that the user can perform game actions in a natural way, similar to how “normal” games work. There will be no confirmation dialogs bugging users for each action he takes in a game, as seen in Ethereum MetaMask and EOS Scatter.



test Account
[redacted]
8.208 ETH
2331.81 USD
[redacted]

Amount 0 ETH 0.00 USD
Gas Limit 49903 UNITS
Gas Price 100 GWEI
Max Transaction Fee 0.004990 ETH 1.42 USD
Max Total 0.004990 ETH 1.42 USD
Data included: 100 bytes

However, actions which are sensitive, such as a transfer of a large sum of tokens, might require confirmation using a different sub-account which is managed by ChromaWallet. This means that malicious dapp code won’t be able to do significant harm. This also means that large-value transactions can benefit from 2FA or hardware wallet integration implemented in ChromaWallet.