Plebble: blockchain, local economies, marketplaces and social order

root1m3@plebble.us
April 2023

Abstract

This paper introduces Plebble, an open-source project that aims to address the issues of distributed consensus, local economies, and privacy wallets in the cryptocurrency space. In contrast to many other cryptocurrency networks that are often driven by hype without substance, Plebble is grounded in the values of the cypherpunk movement and seeks to empower individual sovereignty while promoting a social order based on peer-to-peer interactions. We argue that a future of multiple networks underpinning worldwide values requires a compelling global low-cost distributed organism that can progressively replace central governments with effective wealth redistribution without compromising freedoms. We present both philosophical and technical aspects of the Plebble project, which includes a reference implementation and a running mainnet.

1 Introduction

The contemporary human civilization is situated upon the bedrock of ancient structures that were erected countless eons ago. This empirical observation suggests that our current society has emerged as a direct consequence of the cumulative efforts and endeavors of numerous cultures and civilizations that have thrived and perished throughout the expanse of human history. The existence of these edifices is a testament to the remarkable engineering prowess and architectural ingenuity of our predecessors, and bears witness to the enduring legacy of their achievements. The discovery and study of these ancient structures continues to provide valuable insights into the evolution of human civilization and serves as a reminder of our shared heritage as a species.

It is worth noting that the vast majority of these decisions were made well before the advent of technological breakthroughs such as the widespread availability of the internet in the 1990s, the landmark achievement of the first communication between two computers via telephone by Lawrence Roberts in 1965, and the invention of the first electronic general-purpose digital computer, ENIAC, in 1945. This disparity between the pace of technological progress and the comparatively sluggish advancement of our social structures, including government and law, underscores the need for ongoing efforts to bridge the gap between these two domains. Failure to do so risks perpetuating outdated policies and practices that may no longer be effective or relevant in our rapidly evolving technological landscape.

Moreover, it is important to note that changes in our social structures are not only gradual, but also tend to be incremental in nature, involving small course corrections that build upon the existing state of affairs. This can often result in a certain degree of inertia and resistance to change, as well as a tendency to cling to established norms and traditions, even in the face of new challenges and emerging opportunities. However, it is crucial to recognize that such incremental changes can also be a source of stability and continuity, providing a sense of continuity and predictability in an otherwise rapidly changing world. As such, a balanced approach that recognizes both the need for continuity and the imperative of change is essential for ensuring the long-term viability and success of our social structures.

The advent of the internet has fundamentally transformed the way in which we conceive of and interact with our social structures. In many ways, the internet has served as a game changer, enabling us to overcome the constraints of distance and connect with others on a global scale. The ability to communicate and exchange data with anyone, anywhere, at high speed has revolutionized the way in which we conduct business, share knowledge, and build relationships.

However, it is worth noting that our inherited social structures were not designed to accommodate these kinds of technological advancements. Our traditional models of representation and centralized planning are highly territorial in nature, and are often ill-suited to the rapidly changing and highly interconnected world of the internet. As a result, there is a pressing need for us to re-imagine and re-invent our social structures to better reflect the realities of the digital age.

This requires a concerted effort to explore and research new models of governance, collaboration, and decision-making that can better accommodate the realities of the internet era. It also demands a willingness to challenge established norms and assumptions, and to embrace the potential for innovation and change that these new technologies present. Ultimately, only by re-imagining our social structures in light of these transformative technologies can we hope to build a more resilient, adaptive, and inclusive digital society for all.

The central aim of this project is to explore new and innovative models of social order that can potentially improve resource allocation, reduce operational costs, and ultimately enhance the quality of life for individuals across the globe. Specifically, the project is focused on developing a flat organizational structure comprised of numerous small, autonomous entities that are fully sovereign and not dependent on existing government structures.

The rationale for this approach is that traditional government structures have proven to be highly centralized and bureaucratic, often leading to inefficiencies and inequities in the allocation of resources. By contrast, a flat organizational structure characterized by many small, autonomous entities has the potential to distribute power and decision-making more evenly, resulting in a more equitable distribution of resources and a greater degree of responsiveness to the needs and concerns of individuals.

One key advantage of this approach is that it can potentially be less expensive to operate than traditional government structures, as it relies on smaller, more decentralized entities rather than large, centralized bureaucracies. This can free up resources that can be directed towards other priorities, such as infrastructure development, social welfare programs, and environmental sustainability initiatives.

Ultimately, the goal of this project is to develop a more robust and resilient model of social order that can adapt to the rapidly changing technological and social landscape of the 21st century.

Let us embark on a thought experiment and imagine rewinding time all the way back to the era of cavemen, or even before. Let us assume that modern amenities such as light, water, and communications already existed, and that we have now introduced a new resource: computers and the internet. With these technological capabilities as our foundational plumbing, we fast-forward through history to explore how our social structures might have evolved differently.

Would countries, fences, and governments have ever been created in this alternative reality? Would representative democracy, politicians, and voting have even been invented? And when it comes to the exchange of value, would we have replaced bartering with central coins if computerized algorithms, automation, and tokenization had existed at the time? Would we have still developed centralized laws applicable to specific territories?

As we contemplate these questions, it becomes clear that we need to accelerate our evolution and increase our standards of living, knowledge, and freedoms. Our ultimate goal should be to eradicate poverty and create a society in which working for a living is no longer a necessity.

This requires a call to action for cooperative work and collaborative efforts to explore new and innovative models of social organization that can accommodate the realities of the digital age. By harnessing the power of technology and embracing new ways of thinking about governance and decision-making, we can build a more equitable, sustainable, and prosperous future for all.

Plebble represents an initiative to construct a foundation for such infrastructure in a grassroots manner. While we have not yet reached our ultimate goal, let us assume that every person on Earth, including our hypothetical future friends living on Mars and those residing on a futuristic Titan Orbital Station, has access to a reliable computer and high-speed internet connection.

This project encompasses not only technical design details and a reference implementation, but also a philosophical framework for understanding the context and significance of this endeavor. By building a trusted and decentralized network infrastructure, we aim to create a more inclusive and equitable digital society, one that can withstand the challenges of the modern age and promote the flourishing of all individuals and communities.

2 Related work

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending

Since its release in January 2009, *Bitcoin* has established itself as the preeminent cryptocurrency, garnering significant attention from researchers, developers, and enthusiasts alike. Over the years, countless other cryptocurrency protocols have emerged, fueled by novel insights and disruptive aspirations, such as challenging the trust-based institutions of the banking system. However, despite the proliferation of alternative options, Bitcoin remains the dominant protocol for those seeking an alternative global economy, boasting a near-ubiquitous presence in the cryptosphere and leading the marketcap charts [21].

It is important to note, however, that being the first approach, *Bitcoin* is not without flaws and is likely to be subject to improvement. Nonetheless, its enduring influence and popularity demonstrate the enduring appeal of decentralized, trustless digital currencies, and suggest that the cryptocurrency movement is far from over.

Despite *Bitcoin*'s strong network effect designed to be a monopoly [16], numerous initiatives have been explored, either as protocol forks or entirely new developments. Forks typically introduce slight variations to the original protocol, such as NameCoin [19] and LiteCoin [20] (the first and second altcoins, with variations in block speed and hashing algorithm), BCH [15] (variations in maximum block size), Ergon [14] (variations in the rewards algorithm), DogeCoin (variations on social appeal by introducing fun), along with an ever-growing list of *Bitcoin* forks. New developments typically introduce more significant conceptual changes rather than small tweaks, such as Ethereum [17] (tokenization, smart contracts), IOTA [18] (multidimensional DAG instead of a linear blockchain), NEM [19] (an early Bitcoin competitor written in Java rather than C++), NANO [22] (fast, zero-fee transactions), and many more.

Discussions about protocol supremacy are a common and often toxic topic in social media. Unfortunately, they tend to devolve into unproductive debates fueled by hatred and closed-mindedness, with little progress made toward any meaningful resolution.

In order to facilitate a smooth transition into web3, the question arises as to what kind of node software should be developed that would be universally accepted, without any apprehension of making the wrong choice. The solution lies in a peer-to-peer (P2P) protocol that doesn't impose a set of central rules agreed upon by all, thereby allowing individual nodes to run on their own rules without the risk of being invalidated by others. This meta-protocol, without any common rules, forms a crucial aspect of Plebble's vision.

Part I. Philosophy

This section reflects the author's personal perspective, shaped by his experiences, education, and inspirations. While the development process was primarily objective and impartial, it remains open to incorporating other ideals, as detailed in Part II.

1 Vision

Although the idea of replaying history may seem daunting and difficult to execute, the challenge lies in the fact that every time we override a decision, we effectively fork the timeline, and this process applies to every decision made in the past without considering the internet. However, it is the final fork resulting from the composition of all the forks that we are interested in pursuing.

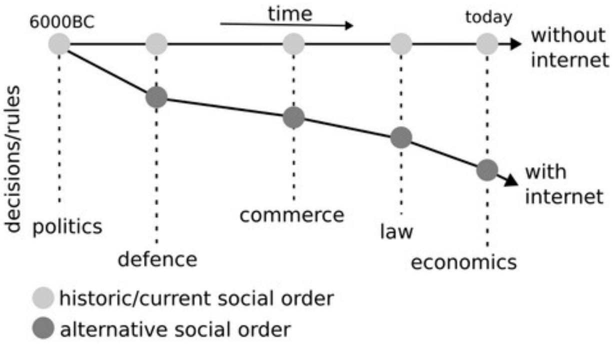


Figure 1: Illustration of the history-fork idea based on the existence of instant P2P communnications since the beginning of human civilization.

Even though we have not followed the procedure formally, we strongly believe that it serves to illustrate the idea of proposing a change that does not rely on adding incremental modifications to our current state, but instead on a process of rebuilding the foundations of our society to discover an alternative model that we could transition to.

It is possible to imagine a society in the future that is not bound by the structures inherited from a time when fast and instant peer-to-peer interactions were not possible. We are motivated by the belief that it is worthwhile to invest our time and effort in joining the movement and developing a prospective technology that can serve as the seed for the production of fundamental plumbing infrastructure suitable for creating a new social order that can accelerate our evolution as a civilization [1].

Our vision is built on a foundation of peer-to-peer networks. We envision a future where everyone actively participates by running their own node, with a flexible range of capacity and power. This includes the use of nodes that are as affordable as raspberry pi, which minimizes user costs and reduces entry barriers.

2 Economics.

Inflation/deflation.

The popular saying in the Bitcoin community, "Be your own Bank", emphasizes the idea that we can operate our money without relying on third-party institutions, particularly banks. However, this perspective does not consider the larger picture of the economy, including the entire cycle of money, from creation to destruction.

Bitcoin allows for the operation of money without intermediary institutions, and as a global currency, having a fixed monetary supply is a sufficient proposition. The protocol, enforced by participant nodes, ensures that no one can influence the rules of money creation because they are set in stone.

Printing money.

The idea of a global coin can give the impression of centralization, as it implies that all values are forced to be represented under the same monetary scheme. This means that all values are linked and affected by each other, rather than being disconnected. This interconnectedness can lead to the propagation of waves and, eventually, the production of tsunamis, which can be seen as a metaphor for global crises.

At a certain level of abstraction, both Bitcoin and central banks have similarities. Both are responsible for the creation and destruction of global money employing a specific set of rules or procedures. They both are systems that solve an optimization problem in different ways. By controlling the amount in circulation they are designed to maximize social efficiency.

Central Banks and Bitcoin, through their algorithms, control the amount of money in circulation to maximize social efficiency. When there is too much money in circulation, people may stop producing and trading value, and shortages can also have the same effect. Therefore, there exists an optimum amount of money in circulation that varies over time (Bitcoin is deflationary and supply never increases), enabling people to focus on delivering value to others, which can be referred to as *social efficiency*.

As global money represents the aggregation of all possible values, it is difficult to determine the optimal point where social efficiency is achieved. All values are approximate and relate to macroeconomics.

As mentioned, Bitcoin runs a deflationary model. A limited supply of 21M ensures zero inflation. However supply experience some reduction caused by different events: Over time users lose their private keys, or never move them (e.g. approximately the first million Bitcoins have never been moved by Satoshi Nakamoto and could potentially be forever lost). Other users intentionally burn tokens by transferring coins to arbitrary addresses with unknown secret key for hash timestamping purposes.

We find the Bitcoin cryptoeconomics too simple to be considered beyond the realm of a first approximation to a more precise real world scenario aiming to optimise social efficiency.

Be your own central bank.

We put forward the lema “be your own **central** bank”, a twist on the Bitcoin lema “Be your own bank”. The later meaning that you own and control your secret keys without the need of a supervising authority.

A Global Coin, either Bitcoin or Fiat, carries a notion of centralization by representing all possible values humans can create and trade with, an idea usually represented by the symbol ∞/21M.

Being your own central bank neutralizes such conceptual reduction that limits resolution and forces an inaccurate representation of reality. Under this lema, in addition to owning coins, individuals also own the right to control their own coin inflation rules, idea that can be represented by the symbol ∞/∞.

The concept boils down to distributing the economy as if there were as many Central Banks issuing different coins as people.

Global money, whether it be through Bitcoin or Central Banks, requires the concept of prohibition to be enforced. However, the methods of enforcement differ between the two. Bitcoin achieves it algorithmically by limiting the creation of new Bitcoins, while Central Banks rely on legal regulations to prevent the printing of money outside of certain parameters.

And, naturally, both models exhibit centralization when they are operated as a monopoly in absence of competition. In Bitcoin this is known as *maximalism*. Fiat is controlled centrally in all respects, while Bitcoin's monetary rules are central, enforcement is only distributed. A global coin, be it Fiat or Bitcoin, also promotes speculation, bubbles, and crashes.

Let us embrace complete decentralization. Under the motto "Be your own central bank," we introduce the concept that individuals can create and destroy their own money, enabling them to digitally trade any value they can contribute to society.

Let's empower individuals with control over inflationary rules and exchange prices, utilizing automation to simplify complexities and allowing anyone to actively participate in the global economy by operating their own coins, similar to how Central Banks operate state money. This "Be your own central bank" approach grants individuals freedom, but also comes with the responsibility to avoid creating scam-coins, even though they may emerge. However, as long as their presence does not negatively affect others, they should be allowed to exist. A scam-coin is one that fails to deliver its associated promise, while an honest coin does fulfill its promise. Exchange operation is what determines the trustworthiness of a coin, and algorithms can rank them accordingly.

The true measure of scarcity lies in a coin's ability to fulfill its promise, not in its monetary supply, which in the case of cryptocurrencies is not possible to guarantee anyway. Trying to enforce artificial scarcity through algorithms is futile. While Bitcoin has artificially limited its supply to 21 million coins, this approach assumes a maximalist perspective as if the Bitcoin protocol was ubiquitous and there were no other networks handling global values. This maximalist view leads to a centralized symmetry, which contradicts the idea of full decentralization. In a decentralized world with billions of coins and distributed inflation rules, true scarcity will be determined by a coin's ability to deliver on its promise through the exchange operated by the user.

In this system, there will be a spectrum of coins, ranging from those with fixed supply to those with hyperinflation, and everything in between. The important thing is that each coin has a specific scope and is weakly coupled with the rest of the coins. In other words, each coin represents a local value and does not affect the overall distributed soup of values. Our algorithms will continuously learn and determine the reliability of each coin, allowing us to trade with confidence.

The concept of providing immediate liquidity to any product, service, or commitment is a powerful one. With the ability to create a coin that represents a promise, anyone could start anew by formulating that promise and entering the trading market.

Barter and money.

Debt and barter were the primary systems of exchange before the invention of money. In small communities, the issue of the "double coincidence of wants" was not a problem as it was easy to find a suitable trading partner [23]. However, with the growth of trade and commerce, barter became limited due to the difficulty of finding a trading partner and determining the value of goods and services being exchanged. Money was developed as a standardized medium of exchange to solve these problems. Money allowed for more efficient and effective trade, as it had a widely accepted value and could be used to purchase a range of goods and services. It also made it easier to store wealth and conduct transactions over time and distance. The adoption of money played a significant role in the development of complex economies and modern societies.

Barter revisited.

It is true that tokenization and matching algorithms can help address some of the issues with barter, such as finding a suitable trading partner and exchanging goods and services of equal value. By tokenizing goods and services and using matching algorithms, individuals and businesses can trade with each other more easily and efficiently, without the need for a direct exchange of goods and services.

Tokenization and matching algorithms can address the computational challenges of finding a suitable trading partner and exchanging goods and services of equal value that were previously limitations of barter. With the use of these algorithms, goods and services can be represented as tokens or digital assets, making them easily transferable and tradable. Matching algorithms can then be used to pair buyers and sellers based on their preferences and the tokens they hold, enabling efficient and accurate exchanges.

Coins play a critical role in providing liquidity and divisibility to goods, services, and promises. Considering that coins are indeed medium of exchange intervening in a transaction the name barter might not be exact, since *barter* implies the absence of any medium of exchange.

When we consider the distance between tokens and the real-life assets they represent, the concept of bartering becomes aligned with **local coins**. This is because each token has a one-to-one relationship with a specific asset.

A single global currency (one unique coin) reduces a diverse range of values into one liquidity pool, leading to a centralized and interconnected global economy that is vulnerable to spreading economic shocks and waves, such as growth and recession periods. The 2008 Recession is a prime example of this, as it originated from excessive mortgage lending to high-risk borrowers in the U.S. and then spread to affect economies around the world, leading to a prolonged period of economic hardship for many.

In a distributed economic paradigm, our understanding involves having a proportional number of tokens (including fungible coins and non-fungible tokens or NFTs) as there are values, such are goods, services, and promises. This creates targeted liquidity and facilitates automated trading.

Just like traditional currency, local coins can be utilized to hedge risks and offer investment opportunities for diversifying portfolios and managing wealth.

A useful analogy for comparing global versus local is imagining sailing a boat in open seas and encountering a large wave (like a tsunami) versus encountering many smaller waves. Each wave represents inefficiencies such as market volatility, asset bubbles, financial instability, market crashes, economic downturns, income inequality, or social unrest.

The same type of inefficiencies can be observed in local economic models, but their impact is less severe and more contained compared to global economies. This aligns with the notion of global resilience offered by distributed economic models.

3 Tokenization

To develop an e-barter system, we propose extending the capacity to print money to all individuals, rather than restricting it to national entities. This approach, which we refer to as "being your own central bank," would empower individuals to create personal coins via a sovereign node. Each user would have the freedom to establish their own cryptoeconomic policies, including supply and print/burn politics, along with an associated promise to deliver on the exchange of their coins. This system would resemble the practice of national entities printing and burning money to maximize population performance, but with a more individualized and decentralized approach. Similar to a shop tender, users in the proposed e-barter system would deliver on their promises in exchange for the associated coin, which they issued and distributed at an earlier time. In this context, a coin represents a debt for the creator and acquisition power for other users.

We define functional and reliable coins as those that allow users to successfully exchange them for their associated promise. Any failure to deliver on the promise would result in the flagging of scam coins. We do not fear the creation of scam coins and advocate for their free creation. Rather than enforcing prohibition, punishment, or any form of social cancellation, we propose living safely with them with the aid of our trusted AI. Our approach aims to maximize both freedom and safety in the e-barter system.

The proposed e-barter system may lead to a positive change in users' perception of the economy. Prices may no longer be indicative or useful in cases where a wallet contains thousands or millions of different coin balances. However, users may gain a better understanding of their acquisition power relative to their total wealth.

In this new paradigm, imagine a supermarket built solely for a user's personal use in the metaverse. The user could view all available items, including goods, services, or promises, and purchase items that are "tagged for them" with a price represented as a percentage of their own wealth. As soon as an item is added to their shopping basket, their acquisition power decreases, and products that are no longer within their reach disappear from view. For instance, adding a Lamborghini Diablo to the basket may leave only a loaf of bread and a few poems available for purchase. However, returning the car to the shelf would reveal a full array of products to choose from.

Such an experience is possible with the use of the Plebble wallet, which is equipped with integrated AI.

4 Scarcity

Satoshi Nakamoto created a fixed supply of 21 million Bitcoins to ensure that the coin would be a scarce and valuable resource. However, this design did not account for the possibility of dealing with multiple networks. As legitimate Bitcoin software forks and continuous research and development on the technology result in the emergence of new networks, an increasing number of coins become available to the public, rendering the notion of an infinite cryptocurrency supply. The Bitcoin lema ∞/21m (everything divided by 21 million) is not realistic unless a single network is agreed upon worldwide, a scenario that is unlikely given the continuous nature of innovation.

The idea of replicating the scarcity model of fiat currency in the cryptocurrency space is a misguided approach that has led to maximalism and conflict among enthusiasts vying for the survival of their preferred coin. This has resulted in a toxic atmosphere that undermines legitimate research and experimentation aimed at finding better solutions for building a more trustworthy society. Unlike in the traditional world, where nations use the law to control currency scarcity and only allow their trusted central banks to print money, such an approach is unfeasible in the cryptocurrency space. Any design that seeks to limit the supply of cryptocurrency is inherently flawed. Instead of replicating the fiat model, it is essential to explore alternative solutions that are better suited to the unique features of the cryptocurrency space. By doing so, we can foster a more collaborative and productive environment that is conducive to building a more secure and efficient financial system.

To design a successful cryptocurrency, it is necessary to consider the co-existence with numerous other crypto platforms, potentially millions of them. Plebble's approach to this challenge is to encourage personal coin creation, allowing individuals to centrally control the printing and burning of their own money, similar to central banks. This approach enables maximum distribution and leverages on the unavoidable fact of an infinite supply, with scarcity not residing in the coin itself, but rather in the ability to deliver on the associated promise. In contrast to traditional currencies, where scarcity is enforced through laws and regulations, in the cryptocurrency world, scarcity is a function of built trust through the ability to fulfill promises made through the use of personal coins. By prioritizing the ability to deliver on promises over artificial scarcity, Plebble aims to create a more robust and equitable cryptocurrency ecosystem that can better serve the needs of individuals and society as a whole.

To provide a concrete example of the concept, let us consider the case of Alice, an individual who creates a cryptocurrency called bikeCoin to facilitate the sale of bicycles in her business.

The promise underlying bikeCoin is simple: it can be used to purchase bicycles from Alice's shop. As the creator of bikeCoin, Alice has the authority to control its supply, continuously adjusting the amount of currency in circulation through the creation or destruction of coins. Alice would employ an algorithm that takes into account various economic parameters in order to maximize sales. When the demand for bicycles outstrips the supply of bikeCoins, Alice would burn bikeCoins (or increase prices) to reduce demand. Conversely, if unsold bicycles accumulate, she would print more bikeCoins and put them into circulation (or lower prices). Through such mechanisms, Alice has complete control over her local economy. The perceived value of bikeCoin will be tied to Alice's ability to honor her promise to buyers, thereby ensuring a smooth exchange process.

5 Politics.

The origins of representative democracy can be traced back to ancient Greece and Rome. In Athens, citizens would gather in the agora (public square) to discuss and debate important issues facing their city-state. This system of direct democracy allowed citizens to participate in the decision-making process and vote on matters such as war and peace, taxation, and public works projects.

In Rome, the concept of representative democracy was developed. Citizens elected representatives to make decisions on their behalf, rather than participating directly in the decision-making process. This system of governance was more efficient and allowed for the creation of a professional class of politicians who could devote their time to governing.

Representative democracy as we know it today developed in Europe during the Middle Ages, particularly in England. The Magna Carta, signed in 1215, established the principle that the king was not above the law and that certain rights were guaranteed to all citizens. Over time, the power of the monarchy was gradually curtailed and a system of parliamentary democracy emerged, with elected representatives making decisions on behalf of their constituents.

The ideas of Enlightenment thinkers such as John Locke, Montesquieu, and Rousseau also had a significant impact on the development of representative democracy. They argued that governments should be based on the consent of the governed and that power should be separated into different branches to prevent tyranny. These ideas influenced the drafting of constitutions and the establishment of democratic governments around the world.

It could be argued that if there had been a technology available at the time that was designed to circumvent the need for third parties, the evolution that took place may not have occurred.

In the present model, decision-making power originates from the people but is then transferred to a relatively small number of groups through the voting mechanism. This transfer of power leads to the people losing their decision-making ability and being excluded from the process until the next election cycle. Additionally, public demonstrations may arise as a result of this loss of power, which could have been avoided if people had not been sidelined in the first place.

The voting process takes the diverse perspectives of individuals and reduces them to a few select ideologies that are deemed to be the best fit. This can result in the unique ideas and values of individuals being overshadowed by the dominant ideologies that are represented in the voting process.

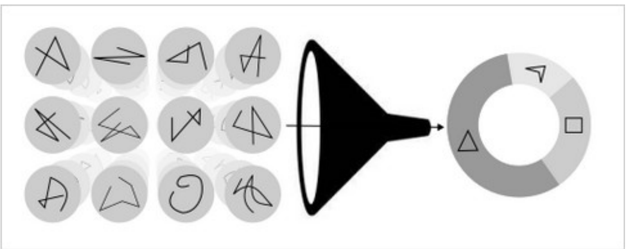


Figure 2: The voting process reduces rich individual mindsets into a contribution to the best matching of a handful ideologies.

The accumulation of power often leads to centralization, which in turn creates a vulnerability to corruption. As power becomes concentrated in the hands of a few, there is an increased risk of abuses of power and unethical behavior, as there are fewer checks and balances to prevent such actions. This centralization can be particularly susceptible to corruption, as those in power may become more interested in maintaining their own power and influence than in serving the interests of the wider population.

With the appropriate peer-to-peer technology, it may be possible to establish and maintain a public system where decision-making power is distributed among all individuals. Such a system could allow every person to have a voice in the process, steering the system according to their own interests at any given time. This would be possible because the common interest of the population is merely the sum total of each individual's interests. As a result, traditional political structures such as voting theatres, representatives, and politicians could become obsolete, replaced by user automation that acts on behalf of individuals with maximum accuracy. By removing the need for third parties in the political process, this system could potentially be more efficient, transparent, and responsive to the needs of the population.

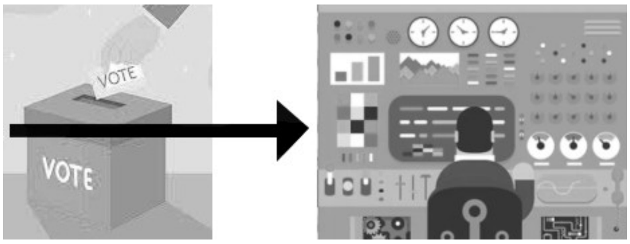


Figure 3: Ballot boxes replaced by control panels.

In this proposed system, the traditional ballot boxes used in elections could be replaced by control panels. These control panels would allow individuals to have direct control over the decision-making process, enabling them to make informed choices and have their voices heard. By replacing the physical act of voting with digital control panels, the process could be streamlined and made more accessible, potentially increasing participation and engagement among the population.

6 Law.

We the People
insure domestic Tranquility, provide for the common
and our Posterity, do ordain and establish this Constitu
Article I

In light of our focus on decentralization, it is imperative to approach the concept of law from a perspective that critically examines the prevailing centralized structure and proposes a framework for a peer-to-peer decentralized legal system.

One of the primary characteristics of contemporary law is its disregard for individual sovereignty, as no individual holds authority over the prevailing and formidable system that enforces a social construct of regulations that apply to all members. However, upon closer examination, this notion is debatable. Various laws exist that differ from one another and are specific to people residing within particular boundaries. While this may be viewed as a decentralized approach, it remains rather centralized as a large number of individuals are subject to a shared set of rules irrespective of their agreement with them.

Law is a complex subject, and any attempt to revamp it has the potential to spark considerable controversy. Nonetheless, in the realm of research exploring alternatives, it is imperative to remain open-minded and receptive to new ideas.

To streamline the discussion, it may be helpful to differentiate between actions performed by individuals utilizing digital means, naming it *digital law*, and other actions occurring in contexts beyond the scope of this analysis.

Although they may be regarded as outlaws within the current establishment, as sovereign nodes, they possess a system for dispensing justice.

Nodes have independently developed a law in a specific electronic exchangeable format. Unlike the traditional concept of law, such regulations are created by the nodes themselves, and consist of a collection of articles that they have either created or borrowed from prior interactions, reflecting the conditions they are prepared to present and implement in each peer-to-peer interaction with other nodes.

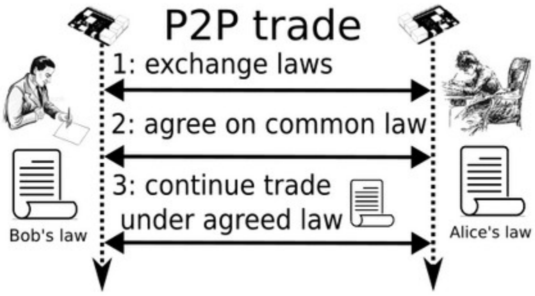


Figure 4: Common law is obtained after a merge process between two laws supplied by each peer. The outcome, when signed by both parties, serve as regulatory basis for the remaining events in the interaction.

In a peer-to-peer interaction between two sovereign nodes, they can exchange their respective laws, scrutinize each other's articles, and potentially integrate those articles from the peer's law that would enhance their own regulations. Subsequently, both parties negotiate and settle on a mutually acceptable law that will govern their future course of events.

Undoubtedly, this scheme should be automated to the greatest extent possible as relying on human intervention alone could result in an unmanageable overhead.

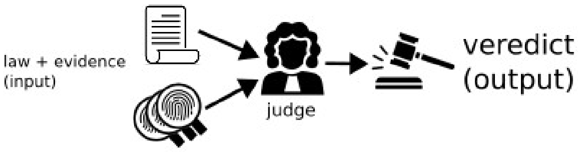


Figure 5: The elements involved in an automated justice systems are:

Evidence: Verifiable facts.
Article: program to score evidences.
Law: collection of articles.

Judge: program that processes evidences against a given law to produce verdict.

Verdict: penalty/compensation result

A law is essentially a compilation of articles, each of which constitutes an interchangeable unit of information and code, forming a program. In addition to textual content in multiple human languages, each article also contains code that generates a score or percentage of compliance when provided with evidence. Another program, known as a judge, considers this score to evaluate the entire situation and provide a verdict.

This process can be executed in real-time during any peer-to-peer transaction to provide continuous feedback on whether the agreed-upon terms are being followed. This can be helpful in adjusting behavior promptly to prevent any issues from escalating.

The fundamental system can be linked to a penalty/compensation contract that could be automatically executed based on the judge's verdict.

Law is code. Code is Law.

The idea of a decentralized justice system that can operate almost entirely in an automated fashion, enabling individuals to keep their interactions in line with the agreed-upon terms in real-time, has the potential to significantly reduce the burden on the traditional justice system. This is not only due to the potential increase in the number of verdicts that can be efficiently delivered by leveraging advancing technology, but also because it could minimize the number of cases that escalate simply because of a lack of awareness by one or both parties.

7 Anonymity. Identity.

Discrimination continues to be a pressing issue, and a considerable amount of tax-payer money is spent on fighting it. However, the fundamental concept of identity, which reveals sensitive personal information such as race, age, gender, and skin color, serves as the basis for states and regulated businesses. Such operations are reliant on the 1-to-1 association of person-id, thus opening the door for discrimination on various logic levels.

The concept of identity is central to the operation of states and regulated businesses. Personal identification, such as national identification numbers, drivers licenses, and passports, serves as the basis for accessing essential services and participating in society. Such identification facilitates the collection of sensitive personal information, which can be used to discriminate against individuals based on their characteristics.

The 1-to-1 association of person-id also serves as the basis for government and business operations, enabling the identification and tracking of individuals' activities. However, this association can lead to discrimination on various logic levels. For instance, biased algorithms and automated decision-making systems may utilize personal identification data to make decisions that disadvantage certain groups, such as those of a specific race or gender.

Moreover, the use of personal identification data in law enforcement activities can lead to racial profiling and discrimination against specific groups. Additionally, the reliance on person-id in the workplace can result in discrimination in hiring, promotion, and salary allocation.

Discrimination remains a pervasive problem in society, and addressing it requires a fundamental rethinking of the role of identity in the framework of states and regulated businesses.

It is common practice in literature to use pseudonyms, particularly when the content is dissident in nature. This practice dates back centuries and has been utilized by many authors. For example, José Augusto Trinidad Martínez Ruiz, who was a political radical in the 1890s, used the pseudonym "Azorin" to sign his works. Similarly, in 1721, Benjamin Franklin was denied the opportunity to publish a letter in a newspaper and resorted to using the pseudonym "Silence Dogood".

In contemporary times, using monikers, nicks, aliases, and avatars to register social media accounts is widely accepted as a normal practice for safety reasons. Revealing one's true identity online can be risky, especially for vulnerable and impressionable children.

However, it is worth noting that governments can also benefit from using individuals' true identities, particularly in law enforcement and crime investigation. Techniques such as follow-the-money and identity tracking can be more effective in catching named criminals. This aligns with the common trade-off where individuals may sacrifice certain freedoms in exchange for greater safety, a notion that is often endorsed by governments.

In our thought experiment about re-making the system, we have temporarily set aside the issue of crime fighting, including how illegal individuals would be prosecuted. Instead, we have focused on designing a system for honest people (the happy case). This decision was made because we recognize that the notion of illegality may differ between a classical law system of any country and our projected re-make of a peer-to-peer (P2P) law system for sovereign individuals. In such a system, individuals would have the freedom to decide how much privacy they are willing to sacrifice in exchange for their own safety, without coercion. We plan to address the issue of physical world crime and how to deal with illegal individuals at a later stage, once we have a clearer understanding of the similarities and discrepancies between the two legal systems.

Personality

In our proposed system, we recognize the need for an intermediary artifact between an individual's true identity and their digital presence. This intermediary artifact is what we refer to as personality. Personalities are anonymous digital identities that individuals create and use to interact with society. The link between an individual's true identity and the personality they create is intended to be kept private and never disclosed to anyone unless voluntarily shared. For instance, users may choose to reveal their personalities to governments for record-keeping purposes. We suggest that this process could be automated and integrated into user wallets for those who opt-in to this feature.

Users of the proposed system, called Plebble, have the ability to create an unlimited number of personalities. Each personality is essentially a pair of keys - a secret key and a public key - derived from standard public key cryptography (specifically, the ECC secp256k1 algorithm). The public key is referred to as the personality-public-key and its RIPEMD160 hash serves as the personality-id. The personality-id can be utilized to continuously build user profiles based on P2P interactions, similar to systems that rely on identity-ids. Meanwhile, personality-public-keys can be used for various functions such as signing, verifying, encrypting, decrypting documents, and producing certificates [4].

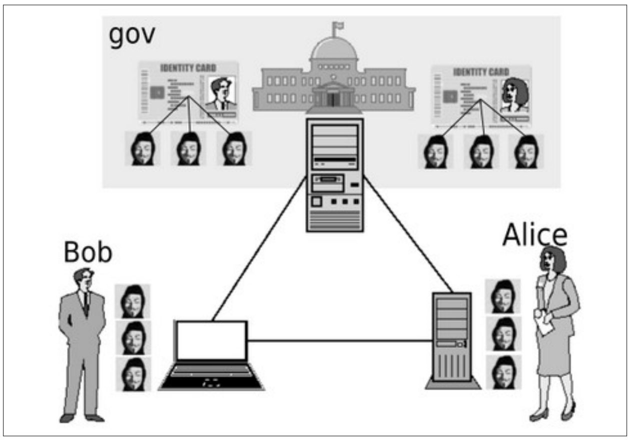


Figure 6: Alice and Bob are able to use the anonymous personalities they have created to interact with each other without revealing their true identities. However, if they are subject to a central government, they may be required to privately disclose their personalities in order to ensure accountability.

Certificates serve as the fundamental building block for constructing webs of trust in the proposed system. These tamper-proof graph structures are secured by digital signatures and can be used by businesses, organizations, supply chains, or any other construct based on trust. Certificates can be leveraged to create automated or semi-automated workflows, allowing for more efficient and secure processes.

8 Financial freedom.

One key feature that we aim to enforce for sovereign nodes is financial freedom. This stands in contrast to financial traceability, which entails the involuntary disclosure of individuals' value exchange activities, ultimately giving rise to a surveillance state. Our proposed system seeks to uphold financial privacy and empower individuals to control their own financial affairs.

While our proposed system aims to prioritize financial freedom and privacy, we recognize that some users may choose to use submissive wallets that report to a government via a design-for-purpose intrusive plugin. In such cases, the wallet would still allow for private communication with the government to disclose any required details about the user's financial affairs, allowing the government to carry out its accountability activities. However, it is important to note that this would be a choice made by the user, and not a mandatory requirement of the fundamental system, which is unaware of the country or government the node is operating. However, governments may choose to distribute a version of the wallet software that is equipped with the appropriate compliance plugin.

Regardless of whether a node is sovereign or submissive, our proposed system seeks to ensure that no public trace of any financial activity is left that could lead to the disclosure of a user's economic activity. Financial privacy is a fundamental principle of our system, and we aim to design and implement it in a way that prioritizes individual autonomy and control over one's own financial affairs.

9 Individual Sovereignty

The flat organization of nodes in our proposed system can be viewed as a resilient global system that promotes a form of social order, where every node has sovereign responsibilities, similar to how nations operate today. This design philosophy potentially transforms centralized planning into a self-managed society that encourages individual sovereignty and equal distribution of power among all nodes, rather than a centralized approach that emphasizes control and hierarchy. This decentralized approach offers numerous benefits, including the creation of a resilient system that is capable of withstanding attacks and disruptions, as well as fostering greater collaboration and cooperation among participants. At maximum distribution configuration, our proposed system has the potential to create a thought size of 10 Billion small individual nations, rather than a single social order for the world, which would promote a more diverse and decentralized society.

10 Egalitarianism

The present study proposes that a distributed system, such as Plebble, can be viewed as a global low-cost profit-making machine operated by a public business, where all participants are shareholders. This system serves to effectively execute mass-consumption services, including value-transfer, for the benefit of humans. By enabling all participants to become shareholders, the system provides economic incentives for participation and fosters a collaborative environment. As a result, the proposed system has the potential to transform the way public businesses operate and offer low-cost and efficient solutions for mass-consumption services.

The design of the reward concept in Bitcoin, which is represented by the coinbase transaction, has been found to be flawed. This is because it primarily incentivizes miners, who represent only a small subset of the network, while leaving the vast majority of participants excluded from the profit distribution. This flaw in the reward concept has been identified as a potential limitation of the Bitcoin system, as it can result in a concentration of wealth and power within a select group of miners. This concentration of power and wealth can ultimately undermine the decentralized nature of the system and erode the trust of its users. Therefore, there is a need for alternative reward mechanisms that promote a more equitable distribution of rewards among all participants in the network.

Our proposed design for a distributed system incorporates a more egalitarian wealth redistribution mechanism by ensuring that all profits made by the machine are redistributed to every single participant. This approach provides a fairer and more equitable reward mechanism, as it ensures that all participants benefit from the system's success, rather than just a select few. By promoting a more equal distribution of rewards, our design encourages greater participation and collaboration among all members of the network, which in turn can enhance the overall efficiency and effectiveness of the system.

11 Wealth distribution. Poverty.

In short, a suitable technology infrastructure could involve an inexpensive automated public system that uses a distributed network to enable widespread services such as value exchange. Participants pool their computing resources to enable this system, and profits are shared among all participants. If this system becomes widely adopted, it could lead to the realization of a Universal Salary concept, at least in principle.

The implementation of such a public service could indeed be a potential solution to address poverty and contribute to a more flourishing civilization. By providing a universal salary or basic income through the redistribution of profits generated by the distributed machine, individuals would have the financial means to participate and contribute their talents to society without worrying about basic needs such as food, shelter, and healthcare. This could lead to a more equitable society where individuals are able to pursue their passions and ideas without being held back by financial constraints.

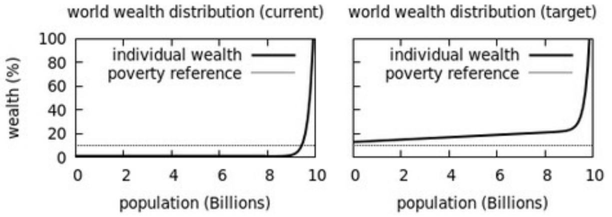


Figure 7: Current and target (desired) models of wealth distribution. Illustrative. Every human being has their position in the X axis with their relative acquisiton power on Y.

To achieve readiness status, which means being ready to positively contribute to society, individuals must have access to fundamental necessities such as a home, heat, health, food, computer, and broadband. If any of these necessities are missing, participants cannot maintain their readiness status and may become a burden as they must focus on addressing personal needs rather than contributing to solving larger societal issues. This is particularly relevant as we face significant challenges as a species, including the threat of extinction from sources like asteroids, pandemics, plagues, and human suffering. Therefore, ensuring that individuals have access to these basic needs is essential to maximizing their potential to contribute to society's greater good.

While promoting financial freedom and encouraging everyone to become economically prosperous by servicing others is important, it is also essential to have a social safety net that does not rely on violent schemes like taxation. A system that solely relies on taxation can potentially lead to unequal distribution of resources and may not fully support those who need it the most. It is crucial to design a system that can provide a social safety net for all individuals, regardless of their financial status, in a way that does not rely on coercive measures. By promoting a system that supports the well-being of all individuals, we can help ensure a more equitable and just society.

This safety-net can be built on the basis of distribution of profits collected from transaction fees among all participants, constituting a spontaneous form of universal salary. If only financial services, or in general services covering basic rights like value-transfers, are carried out in such a way it would help to raise the poverty threshold, perhaps up to readyness levels.

Creating a system that can be operated with low-cost nodes is crucial to enable easy onboarding for individuals who may be experiencing financial difficulties.

12 Liquid economy. Streams.

Money, as a fundamental unit of modern economies, has been studied extensively from different perspectives. In this paper, we explore two distinct models of money: the fungibility model and the fluid dynamics model. The fungibility model views money as an aggregation of indistinguishable units of account, while the fluid dynamics model treats money as a liquid governed by fluid dynamics laws.

Fungibility Model

The fungibility model of money assumes that money can be viewed as an aggregation of indistinguishable units of account, such as 1 sat or 1 cent. This model enables value exchange through transfers of fixed amounts between accounts. Under the fungibility model, each unit of account is interchangeable with another, allowing for seamless and efficient transactions. This model has been widely adopted in traditional financial systems and serves as the backbone of most modern economies.

Fluid Dynamics Model

Alternatively, money can also be modeled as a liquid, where the principles of fluid dynamics govern its behavior. This model treats money as a continuous flow, allowing for dynamic and flexible transactions. A powerful programmable money system can be designed based on streams, where money flows from one account to another based on predetermined rules. The fluid dynamics model of money enables the creation of complex financial instruments and the automation of financial transactions, which can enhance efficiency and flexibility in building complex economic circuits.

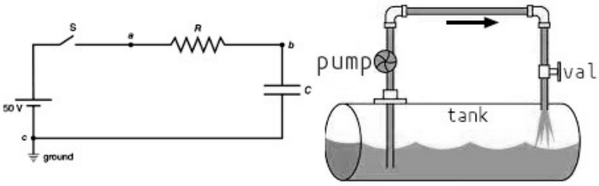


Figure 8: Fluid money modeled as either electrical current (electrons being unit of transfer) or incompressible water (units of volume).

Both models have been used to design effective financial systems, and their combination may offer even greater potential for the future of money.

13 Decentralization.

Decentralization is a general approach that can provide resilience, prevent single points of failure, and enhance censorship resistance. This strategy can be employed in various domains, such as finance, computing, governance, and energy, among others.

When considering network topology, the goal is to create distributed systems that consist of as many independent nodes as possible. Similarly, when it comes to governance, the objective is to have numerous independent decision-making entities. This is particularly important when implementing software upgrades. The human brain can serve as an example of a distributed system, with interconnected neurons working in a control loop. These neurons continually read and respond to external stimuli. A single neuron's failure does not result in brain failure, and the decision-making of an individual neuron does not necessarily reflect the final response of the brain.

The idea of a single global coin implies a sense of centrality, as it suggests that this coin would be used to account for all values in the world. However, this approach oversimplifies the diversity of values that exist, and many values would be misrepresented or overshadowed by others. Instead, we propose a formula where each coin represents a single value. For billions of values, we would use billions of coins, thereby avoiding the problems that arise from reducing all values to a single currency.

Our objective is to achieve the highest possible distribution across all aspects, including geographical, governance, control, down to node resolution, and even software update production and deployment. We also aim to incorporate cryptoeconomics, with the inclusion of personal central banks and local coins, to further enhance decentralization and promote a more equitable distribution of resources.

Our approach is to leverage the insights and knowledge gained from research studies in various fields related to distributed computing. These include studies on P2P networks and routing protocols [58-67], swarm intelligence in robotics and IoT [12], as well as consensus algorithms such as BFT, PoW, PoS, and others. By incorporating these learnings, we aim to design a more robust and resilient system that can effectively handle the challenges associated with decentralized networks.

Swarm networks, which take inspiration from the behavior of various animal species such as flocks of birds, schools of fish, armies of ants, or swarms of bees, have numerous applications across a range of fields, including economics, robotics, self-managed societies, distributed governance, and more. These networks are particularly valuable for their unique properties, which include resilience against predators, environmental factors, and individual failures. By incorporating these principles into our design, we aim to create a more robust and adaptive system that can effectively handle the challenges associated with decentralized networks.

In designing our research protocols, we draw inspiration from concepts that prioritize individual autonomy and sovereignty. These concepts recognize that individuals are, first and foremost, anonymous and independent agents, and should only enter into voluntary arrangements, such as hierarchical associations or government structures, on their own terms. By prioritizing individual sovereignty, we aim to create a system that is truly decentralized, allowing individuals to participate freely and autonomously, without the need for centralized control or authority.

14 Open system

The Plebble principles, which include The Five Pillars of Open Blockchains [11], emphasize the importance of creating systems that are open, borderless, neutral, censorship resistant, and immutable. Our goal is to design a system that embodies these principles, allowing anyone to participate without the need for vetting, registration, or accountability by a third party. This property is commonly referred to as permissionless, which means that anyone can join and use the system without seeking permission or approval from a central authority.

In the context of blockchain technology, immutability refers to the inability to alter or modify transactions once they have been recorded, thereby preserving the integrity of the data and preventing falsification of the truth. This concept does not necessarily relate to the specific method used to store transactions as a chain of blocks, which has been associated with the Bitcoin design. The correlation between immutability and the use of digital signatures and PoW in the Bitcoin design has led to some confusion, as this approach can become non-scalable as nodes sync times rise over time. In the Plebble system, it is easy, cheap, and fast to create a blockchain. However, it is difficult to change the legitimate blockchain once it has been agreed upon, ensuring the immutability of the data.

15 Cryptoeconomics. Supply cap.

In a future where a multitude of different coins coexist, it becomes apparent that it is impossible to represent the global economy with a total of 21 million tokens, as originally intended in Satoshi's design. The creation of new tokens is unavoidable, unless the entire world population voluntarily agrees to a single system. This is a major conceptual difference between cryptocurrencies and legacy currencies issued by central authorities, where users cannot lawfully print money.

As a result, a deflationary cryptocurrency, with a supply cap or upper limit on the amount of new coins that can be created, is not globally effective in the presence of many more systems. This is a flaw in Satoshi's original vision. It is safe to assume that cryptocurrencies, as a whole family, have unbounded inflation as a natural trait, as the ability to create new tokens is necessary to support a growing economy in the long term.

16 Lightweight nodes

Making simple designs is one of the biggest challenges, but we believe it is crucial for creating a successful and resilient protocol. Many level-1 (L1) blockchain protocols have multiple types of nodes with different responsibilities or roles, such as miners, verification nodes, and SVP clients delegates. However, we aimed to simplify the system by using only one type of node. Additionally, we wanted to allow any user to participate using any computer, regardless of its computing power. The distributed system should be composed of a wide range of configurations and sizes, including RAM, CPU, storage, and bandwidth, with the lowest possible minimum requirements. For example, we used the Raspberry Pi, a tiny computer, as a reference hardware for this project.

> Less is More
>
> – architect Ludwig Mies van der Rohe. 1947

17 Scalability.

Consensus on a common truth is one of the major challenges faced by blockchain networks. Bitcoin's common truth is the utxo set, while Ethereum's is a state vector, both produced by a consensus algorithm. However, current designs have limitations in achieving consensus when the ledger size or number of participants increases, leading to performance or security issues [23].

Accounts.

The traditional blockchain concept includes a database model that includes all used accounts, such as the utxo set. However, this design has limitations when it comes to scalability. The financial activity of an individual living on the other side of the world may not be related to or relevant to my own financial activity, and therefore can be considered as two completely separate and independent histories. In other words, not every participant in the blockchain network needs to have access to all information.

Node resources.

To achieve a network of billions of nodes, it is crucial to design the network in a way that scales in an O(1) basis. This means that every new node added to the network should not increase the overhead on CPU, RAM, or bandwidth for every other existing node. In other words, the growth of the network should not negatively impact its performance or security. This is a significant challenge in designing decentralized systems that can accommodate a large number of participants without compromising their scalability or security.

The following technical discussion addresses these issues.

--End of Part I--

Part II. Design.

1 Overview.

The Plebble system is both a social challenge and a technological development. The subsequent sections will delve into the technical intricacies, design decisions, and patterns underlying the Plebble system.

2 Cryptography.

Plebble uses cryptographic functions that employ the Elliptic Curve *secp256k1* [2], with a replaceable design. This means that once it is no longer considered secure (e.g. due to quantum computing), replacing it with the next secure algorithm should be straightforward as the design is prepared for such maintenance since the beginning.

Plebble also has the capability to support several technologies simultaneously, enabling a gradual transition and user selectability of the preferred algorithm.

It is worth noting that secp256k1 has been protecting *Bitcoin* [3] since 2009 and is still considered secure as of 2023.

Public Key Infrastructure (PKI).

PKI is a system that provides a secure method for exchanging digital information over the internet, using public key cryptography. It is a hierarchical system that involves the use of digital certificates and public key encryption to enable secure communication between parties.

A PKI system typically consists of a Certificate Authority (CA), which issues digital certificates to entities, such as individuals, organizations, and devices. These certificates contain the entity's public key and other information, such as the entity's name and digital signature of the CA.

When two parties communicate, they exchange digital certificates, which are used to establish a secure channel for communication using public key cryptography. This means that each party has a public key and a private key, and the public key can be shared with others, while the private key is kept secret.

The PKI system is widely used for securing online transactions, such as e-commerce, online banking, and secure email communication. It provides a strong level of security and trust, as the authenticity of the digital certificate can be verified by the CA, which is a trusted third-party entity.

It is also a crucial component in the development of verifiable credentials [4], which enable the creation of decentralized trust networks.

3 Processes.

The node system consists of two subsystems, each of which handles public and private affairs separately.

A subsystem consists of two processes that run on their respective operating systems, one for the front-end and one for the back-end.

The primary function of the front-end subsystem is to provide a Human-Machine Interface (HMI), which collects inputs from users through either a Graphical User Interface (GUI) or a command-line text console (CLI). These front-end applications communicate with a remote backend where the business logic is stored. The main responsibilities of the front-end subsystem are to capture human input and display feedback.

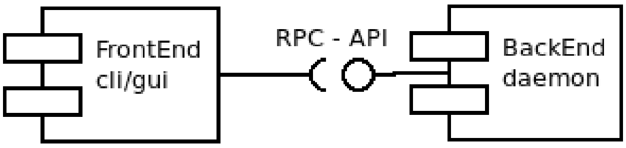


Figure 9: control/display pattern.

The backend daemon process is a resident program that is responsible for executing all the designated tasks that belong to the core business logic. It is controlled by the front-end application through an Application Programming Interface (API) that is implemented over TCP/IP socket connections.

The Plebble node backend is comprised of two separate daemons, each corresponding to a distinct logical view of society: public matters and private matters.

- **gov daemon**. This subsystem contains all the concerns related to Public Matters. Transparency is paramount in the Governance Protocol for the Public System.
- **wallet daemon**. The Private System contains all concerns related to user privacy, data monetization, personal interests, and trade, and emphasizes user sovereignty. The wallet protocol in the Private System places paramount importance on opacity to ensure user privacy.

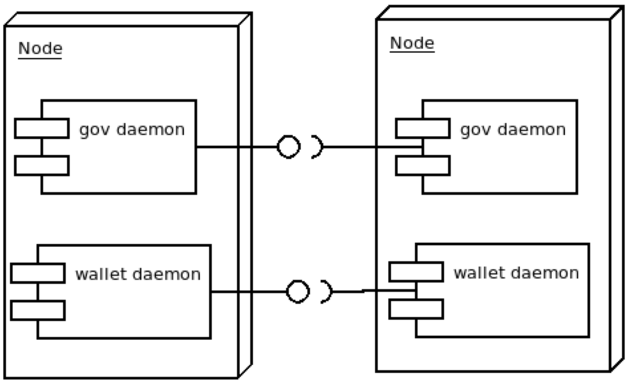


Figure 10: Nodes belonging to different users are part of two distinct P2P networks, namely Gov and Wallet, corresponding to the Public and Private systems, respectively. The daemons, Gov and Wallet, communicate with each other through different protocols. Gov-daemons interact with other gov-daemons via the gov-API, whereas wallet-daemons communicate with other wallet-daemons through the wallet-API. Communication between wallet-daemons is encrypted to ensure the privacy and security of the user's data.

4 API

As part of the design, it is crucial to have a maintainable API specification that facilitates flexible and seamless updates.

Apitool.

Apitool is part of our development tailored for this project. The tool is custom-made API code generator is used to automate API maintenance [53]. It is an API-spec compiler that takes function specs from human editable master files as input [54], producing the following output:

- service numbers, or numeric identification of functions [55].
- API versioning. API version number increments on every network upgrade if needed using detection of API changes during the development since the last upgrade.
- Translation from/to previous API versions for runtime compatibility between nodes implementing different API versions (*svcfish*).
- *C++* and *Java* source files:
  - datagram activity counters for instrumented builds.
  - Svcfish service map include file.
  - RPC call implementations.
  - Implementation prototypes for API functions.
  - Data Transfer Objects (DTO).
  - Message routing implementation.
- *Latex* sources for API documentation.

The configuration scripts in the Plebble system include the integrated Apitool, which automates the process of adding, removing, or editing the API spec. This ensures that updates to the API can be made flexibly and effortlessly without requiring manual intervention from developers, except for implementing new functions or removing deprecated ones.

The tool is extensible by design with the possibility of adding new target languages, specially for developing new front-ends, like WASM, C# or javascript.

All Remote Procedure Call (RPC) function calls are optimized for efficiency using binary encoding in datagrams for serial multiplexed transmission. This design choice avoids any higher-level, human-readable encapsulations such as XML or JSON, which are less efficient.

Developers or users can choose to receive results in text, XML, or JSON formats by specifying the output mode. However, this may incur additional overhead in their front-end computers. Such overhead is usually negligible, and does not have any impact on network communications efficiency.

Plebble specifies two APIs one for each subsystem: **gov-api** and **wallet-api**, which are detailed below.

5 Datagrams.

Plebble utilizes a customized binary protocol that runs over the TCP/UDP/IP protocol through standard sockets for any data transmission between nodes. The transmitted data is known as a datagram, which is a structured data with a simple layout consisting of two sections: a fixed-size header (10 bytes) and a variable-sized payload [27].

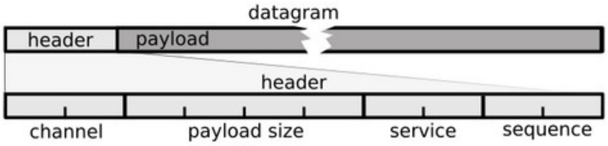


Figure 11: Byte layout and fields of a datagram, or data structure representing bits on wire.

Multibyte fields are all encoded in little-endian.

Channel.

The 2-byte field *channel* is used to allow different networks coexist in the same IP address space.

If two nodes are configured on different channels and try to connect, they would simply fail to establish a connection and disconnect without any further interference.

Special channel numbers are used to define different networks:

- Channel 0: the Plebble mainnet.
- Channel 1: testnet.
- Channel 2: LAN isolated network

Channels 0 and 1 (mainnet and tetnet) are open and permissionless. Channel 2 is designed to be used in development environments.

Other variations such networks with alternative topologies (e.g. ring, star), private networks for organizations and businesses, with or without permissioned nodes controlled by a central authority, all of them would be able to run together in the same IP space by selecting a unique channel number for them.

Service number.

The 2-byte field *service* (svc) consists of 2 bytes and is utilized to classify or assign significance to the trailing *payload* field. Its purpose is to map to a function in the protocol API definition [28], which specifies how the *payload* should be interpreted, including its maximum expected size. For instance, security checks, such as discarding payloads for type "x" that exceed a size of "y," are performed. Additionally, during reception, if the payload exceeds the declared *payload* size (the second field in the header), the connection is terminated, and a separate algorithm is notified. This algorithm is responsible for profiling peers based on their behavior, attempting to determine whether they are malicious or trustworthy.

Sequence.

The field *sequence* (seq) is a 2-byte number used in concurrent API function calls/responses for multiplexation control.

Payload.

Upon the arrival of each datagram, the header is examined as soon as the first 10 bytes are received. Based on this analysis, a decision is made whether to proceed with reading the datagram or drop the connection altogether.

Once the complete datagram is received, additional security analysis is conducted, which includes implementing throttle control to prevent abuse, such as trolling.

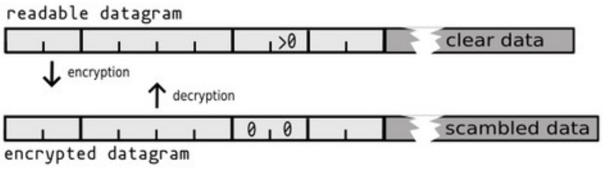


Figure 12: After decrypting an encrypted datagram (svc 0) a readable datagram is obtained.

In the field *service*, the value of 0 has a special significance, indicating that the payload is an encrypted datagram intended for transmission.

Encryption is utilized to protect data from eavesdropping or tampering while it traverses the wire, particularly when the protocol is private, such as in the case of communications between wallets (private systems) or between display and control front-ends and daemons.

The upper logic layers are responsible for determining whether each datagram should be encrypted, depending on various factors such as the service number. For instance, if a datagram is a response to an encrypted call made by the peer in an encrypted form, the response would also be encrypted accordingly.

Encryption is accomplished using the algorithm *AES*(128bit) supplied by libcrypto, with salted rotating keys.

6 Antier. Anonymization overlay.

Datagrams can find their way out to the network via two mechanism configured by the user.

- TCP/IP: datagrams would be delivered immediately, thereby exposing the IP address of the node and making it vulnerable to traffic analysis. This approach provides a moderate level of pseudo-anonymity and is appropriate for users who don't mind publicly declaring that a Plebble node is present on their IP address.
- **Antier**. In this optional mode, datagrams are routed through onion-routing, a technique similar to the one used in the Tor Network, before reaching their destination. This approach offers protection from traffic analysis, node geolocation, and ultimately protects against doxing. It includes features like chaff traffic but comes at the cost of higher overhead in computational costs and transmission delays.

7 Handshake.

Once a connection is established, a verification process is initiated, which involves exchanging three datagrams in a 3-way handshake process.

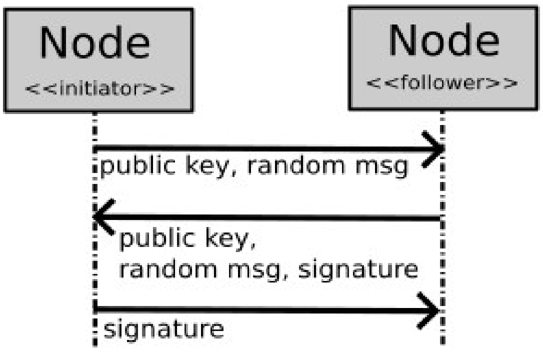


Figure 13: 3-way authentication. Nodes exchange tracking id (public keys) using 3 datagrams.

The handshake process results in the unique identification of the remote peer. This process is cryptographically protected against false impersonation.

As additional information, they also exchange their role and preferences during the verification process. The role is used to determine further logic and can either be a node (an untrusted peer) or a device (a trusted peer for visualization and control, such as a console or GUI app).

In a node-to-node connection, routing tables are exchanged and updated, and the presence of other nodes in the network is spread through routine activities.

8 Addresses/Accounts.

An *address* is a fixed 20-byte wide identifier obtained from a *public key, using* a one-way hash function [10]. The default implementation uses RIPE Message Digest, or RIPEMD160, algorithm [5] as the general purpose hashing algorithm.

Addresses in Plebble are exclusively controlled by individuals in possession of their corresponding private keys. These addresses are used to identify user accounts and private keys grant holders write privileges in the corresponding account.

To avoid typing errors in address input, other systems may encode CRC checks in the final address format. However, this approach can be resource-intensive for the backend and its effectiveness is questionable, as address input errors are not a common occurrence in practice.

base58 encoding has been chosen for encoding addresses in Plebble, primarily because it allows for easy copy and paste operations without the risk of mistyping due to confusing characters. Base58 encoding uses only alphanumeric characters excluding the ones that are easily confused with each other, such as 0 (zero) and O (capital letter), 1 (one), l (lowercase L), etc.

Furthermore, base58 encoding is more compact than base64 encoding which includes special characters, thus reducing the length of the encoded address. This makes it more user-friendly for display in front-end applications and also more efficient for storage and transmission over the network.

Users can create private accounts locally, which can then be shared with other nodes in private communications such as trades. Alternatively, accounts can be shared read-only with the network, allowing for public ledger queries. These accounts serve as generic data containers that can be used for a variety of purposes, including storing the state of Public Algorithms, also known as Smart Contracts in other systems.

Accounts/addresses are managed by the *wallet*. P*rivacy wallets* are components responsible for safe guarding user’s privacy during ongoing *trades*.

The wallet has various responsibilities, including but not limited to crafting, signing, and broadcasting transactions or messages, managing private keys and other secrets such as pictures, writings, or medical records, and keeping track of the user's account balances and transaction history. Additionally, wallets provide functionalities like managing multiple accounts, generating new addresses, and interfacing with different blockchain networks.

Addresses are **Non-Fungible tokens**, broadly known as *NFT. T*hey can contain:

- A special filesystem that uses unique paths and names the content after its hash. This is particularly useful for accessing files with changing content, such as software components distributed via blockchain addresses, trackers, or logs used in IoT and logistics, as well as NFT metadata. The ledger only stores the index, while the actual file contents are stored in a distributed hash table with a 160-bit key space, similar to that of addresses.
- Data maps, or key-value tables.
- Coin balances.
- Connection strings and indirections, e.g. name resolver or a converter that maps a wallet ID to an IPv4 address.

Additionally, *addresses* are **Fungible tokens**, also known as *coins*.

Any user can create *coins* and run local economies with them, just like Central Banks create money for global economies, by simply creating addresses and defining supply and inflation/deflation rules, which can be defined with a program or manually updating the supply with API calls.

It goes without saying that only the individual possessing the private key has control over the associated coin supply or any other associated content.

9 Network topology: cliques and Shards

To achieve optimization for consensus, self-organized groups of 32 nodes called cliques-L0 are formed. These cliques are designed to be sybil-resistant due to their fixed size. As a background activity, cliques engage in consensus rounds by synchronizing their relevant knowledge based on evidence. This is achieved through a signed message passing Byzantine Fault Tolerant (BFT) algorithm, which is particularly efficient for small participant groups.

The responsibility of securing a fraction of the address space is assigned to a specific clique.

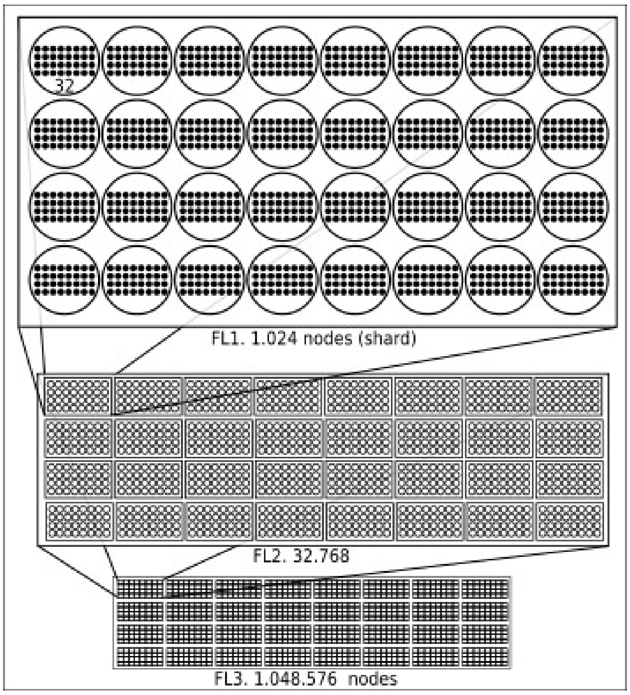


Figure 14: Fractal organizational layout for consensus showing cliques from Fractal Level FL0 (32 nodes) to FL3 (1M nodes). FL6 has a capacity of 34.359.738.368 nodes, ~4.3 nodes per person in the world. FL2 (middle box) contains 32 shards.

We define 7 fractal levels (0 to 6). The first 2 Fractal Levels (0, 1) form *shards*. One shard is then run by 1024 nodes.

10 Filling algorithm

At the start of a new network, there is only one genesis node. However, the system is fully operational in this configuration, with the node actively listening for evidences and constructing a blockchain that encompasses the entire range of addresses (2^160 addresses). Routing tables are empty.

A one-node system is centralized and weak, and is not scalable in terms of throughput or address capacity. Despite its limitations, the one-node system serves as a starting point for decentralization as more nodes join the network.

Once a second computer connects to the network, it is added to a waiting list called the hall. This temporary registry is used to randomly select candidates for configuring the consensus topology. Eventually, the second computer will be chosen and added as the second node to the network.

The process continues as a third node connects to either of the existing two nodes and follows the same process to be added to the network. This process is repeated for the fourth node, and so on, until the 32nd node joins the network.

Once the 32 nodes are connected, FL0 is complete for the first ring of nodes. The next node to join the network will start forming the second ring of nodes.

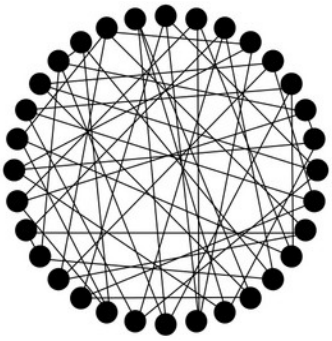


Figure 15: 32 nodes forming one FL0 consensus ring, with an incomplete mesh topology of 3 connections each. They are in charge of building a blockchain based on evidences.

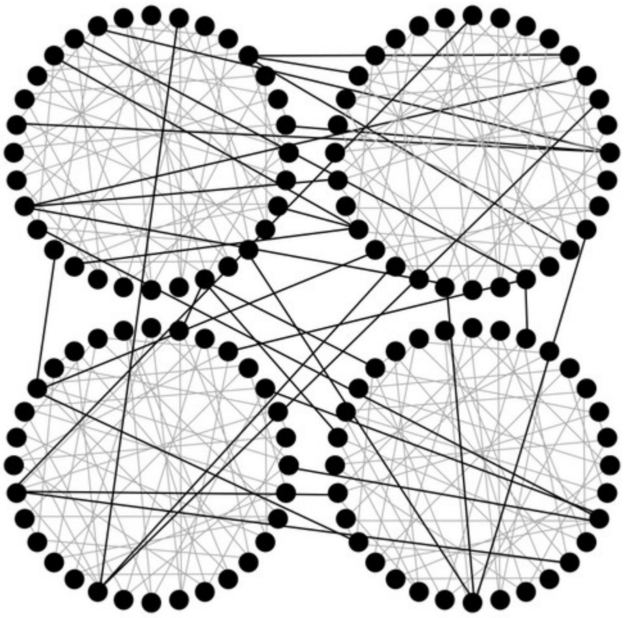


Figure 16: Nodes have connections to other nodes in their L0 clique (gray lines), plus nodes in their L1 clique (black lines), plus nodes in L2, L3, ... L6 cliques (not shown).

The process of adding nodes to the network continues until the 32nd ring is completed. This constitutes the first shard, with a total of 1,024 nodes across 32 rings. Each ring builds a redundant replica of the ledger, resulting in 32 replicas where each node encompass the entire address range.

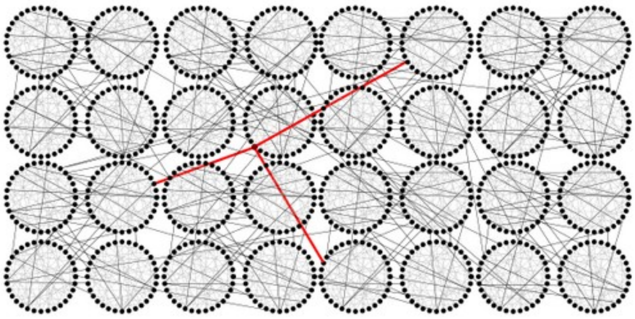


Figure 17: One shard is made of 1.024 nodes organized in 32 identical blockchains (replicas). Each replica is built by 32 nodes agreeing through their FL0 connections (gray lines). Black lines correspond to FL1 connections. FL1 connections of one particular node is shown in red.

The arrival of the 1,025th node marks the start of the second shard and the division of the address space.

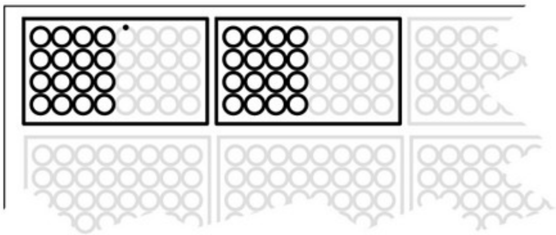


Figure 19: Existing nodes are re-arranged into two shards filled at half capacity. New nodes fill them up before a new split takes place.

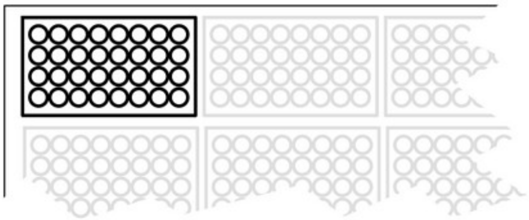


Figure 18: FL2, with a capacity 32 shards, is filled up until completing the first shard with 1.024 nodes arranged in 32 rings). Since at this point there exist only one shard it handles all possible addresses.

At this point, the resulting shards operate at half of their redundancy capacity. The first shard builds redundant blockchains, consisting of 16 complete rings plus one node starting the 17th ring. This shard is responsible for addresses in the first half of the address space, i.e., the addresses in the interval [0, 2^160/2]. It will continue to add 31 more nodes to complete the 17th ring. The second shard is responsible for accounting for the other half of the address space.

As new nodes are continuously added to the network, once a shard is filled, a new reorganization takes place, partitioning the address space into new shards. This process is repeated as shards are filled and then partitioned again.

As each shard is filled, it results in a new split of the address space and the addition of a new active shard. This process continues until FL2 is fully formed with 32 shards, each one managing 1/32 of the address space.

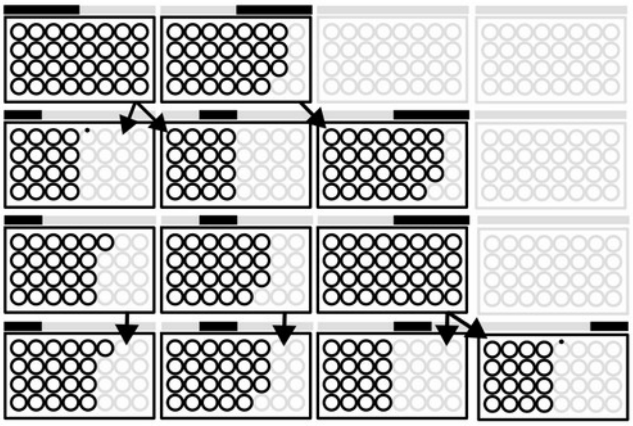


Figure 20: This sequence illustrates the shard splitting process from top to bottom. Each active shard is represented by a black rectangle, with a visual indication above it representing the range of addresses managed by the shard in the form of a black bar.

The process of partitioning the address space and filling fractal levels continues as long as new nodes continue to join. The network expands to cover more and more nodes, reaching up to FL7, where over 10 billion nodes work together to secure the entire address space.

Routing.

To accommodate a target of 10 billion nodes, a 32-dimensional linear tree with a height of 7 is the logical organization. This tree is spontaneously balanced, as the allocation process depends on the random nature of the node public key hash.

It's worth noting that this logical organization doesn't result in centralization because it doesn't involve arranging nodes in layers with a central orchestrator at the top of the hierarchy. Instead, it's a logical search tree that assists in finding the optimal route for transmitting messages to any desired shard.

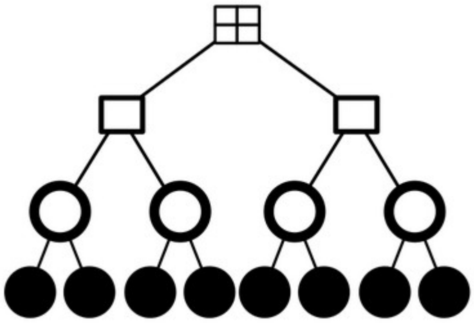


Figure 21: An example of a D-Dimensional tree, also known as a k-d tree [31], is shown in the illustration below. The tree has a dimensionality of D=1, with 2 children (2^1), and a height of 4. Fractal levels (FL) are organized in a similar tree, but with D=5, 32 children (2^5), and a height of 7.

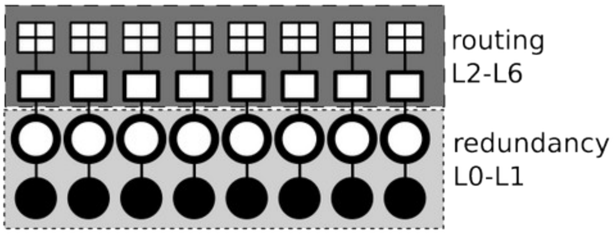


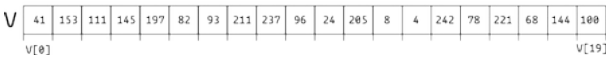
Figure 22: The same tree example (D=2, H=4) demonstrates how each node manages the tree to route messages (from FL2 to FL6) and ensure redundancy (in FL0 and FL1).

11 Encoded coordinates.

The node's public key hash, also known as its address, can serve as a basis for establishing a coordinate system.

This means that the node address or hash can be used to determine the appropriate shard for processing, storing, or retrieving a digital asset. Depending on the number of active fractal levels, it can also be determined whether the asset should be routed for processing by other nodes.

The following calculations will be based on an example node address represented in base-58 encoding as acbxmzYuAhGuHdH2F4NLavDZ5Yo. To obtain a 20-byte vector V representing this address, we can use a base58 decoder.



V	41	153	111	145	197	82	93	211	237	96	24	205	8	4	242	78	221	68	144	100

Figure 23: vector V resulting from base58_decode("acbxmzYuAhGuHdH2F4NLavDZ5Yo");

Scenarios, as network grows in size:

- **FL0**; FL1-6 are inactive; Network size is 32 nodes or less.

All nodes work on a single blockchain (ring 0) accounting for all possible addresses.

The *clique manager*, explained below, would connect to (at least) 3 other nodes picking them from the routing table for FL0.

The concept of *coordinates* doesn't apply in FL0, as every node is in the same *ring*.

- **FL1**; FL2-6 are inactive; Network size is between 32 and 1.024 nodes.

Nodes are organized in numbered <u>*rings*</u> from 0 to 32. In FL1 the number *R* is an agreed number representing the number of active *rings*, and it grows as new *rings* are created on the journey from 32 to 1.024 nodes.

Nodes can determine which ring they are working on by taking the 5 least significant bits of the last byte (V[19]) of their decoded address and taking the modulus of the total number of rings, R.

The modulus operation is used to temporarily allocate nodes to available rings. Nodes that naturally belong to not-yet-enabled rings are temporarily assigned to available rings using the 5 least significant bits of the last byte (V[19]) of their decoded address, which is calculated using the modulus R. If the value of R becomes greater than or equal to V[19]&31, and mod R doesn't result in a cycle, these nodes are re-allocated to their appropriate ring.

For example, let's consider a network with 106 nodes and a value R = 4 (1+floor(106/32)). By convention, the raw value for the ring coordinate is stored in the last byte of the address, which is 100 in our example. To get the last 5 bits of V[19], we need to compute V[19] & 2^5-1, which gives us 100. Finally, we compute the node coordinates using (X mod R), which results in {ring = 0}. Therefore, this node belongs to <u>ring 0</u>.

The *clique manager* uses its routing table for FL0 to connect to nodes in the same ring. The routing table contains other node addresses and helps the clique manager to locate and establish connections with other nodes in the network.

- **FL2**; FL3-6 are inactive; Network size is between 1.024 and 32.768 nodes.

Rings are organized in numbered <u>*shards*</u> from 0 to 32. In FL2 the number *S* is an agreed number representing the number of active *shards*, and it grows as new *shards* are created on the journey from 1.024 to 32.768 nodes.

V[18] contains by convention the raw input for the *shard* coordinate. V[18] = 144**,** and V[19] = 100. Our example node is building a blockchain where e.g. *S* = 13. Then, {*shard*, *ring*} = {(144 & 31) mod 13, 100 & 31}; hence this node has coordinates {*shard*=3, *ring*=4}.

The *clique manager* would connect to (at least) 3 nodes of the same *ring* and other 3 nodes in the same *shard*, picking them from the routing table for FL0 and FL1 respectively.

- **FL3**; FL4-6 are inactive; Network size is between 32.768 and 1.048.576 nodes.

Shards are organized in numbered <u>*knots*</u> from 0 to 32. In FL3 the number *K* is an agreed number representing the number of active *knots*, and it grows as new *knots* are created on the journey from 32.768 to 1.048.576 nodes.

V[17] contains by convention the raw input byte for the knot coordinate, V[17] = 68, V[18] = 144 and V[19] = 100. Our example node is building a blockchain where e.g. *K = 14*.

Assuming FL1 and FL2 are full (no existing nodes have left their rings), if *K* = 14 our node has coordinates {*knot*, *shard*, *ring*} = {(68 & 31) mod 14, (144 & 31) mod 32, (100 & 31) mod 32}; so coordinates are {*knot*=4, *shard*=16, *ring*=4}.

The *clique manager* would connect to (at least) 3 nodes of the same *ring*, plus other 3 nodes in the same *shard*, plus 3 nodes in the same *knot*, picking them from the routing table for FL0, FL1 and FL2 respectively.

- **FL4**; FL5-6 are inactive; Network size is between 1.048.576 and 33.554.432 nodes.

Knots are organized in numbered <u>*clots*</u> from 0 to 32. In FL4 the number *C* is an agreed number representing the number of active *clots*, and it grows as new *clots* are created on the journey from 1.048.576 to 33.554.432 nodes.

V[16] contains by convention the raw input for the *clot* coordinate. V[16] = 221, V[17] = 68, V[18] = 144 and V[19] = 100. Our example node is building a blockchain where e.g. *C* = 7. Assuming again FL0, FL1, FL2 and FL3 are full, if C = 7 our node has coordinates {*clot*, *knot*, *shard*, *ring*} = {(221 & 31) mod 7, (68 & 31) mod 32, (144 & 31) mod 32, (100 & 31) mod 32}; so coordinates are {*clot*, *knot*, *shard*, *ring*} = {1, 4, 16, 4}.

The *clique manager* would connect to (at least) 3 nodes of the same *ring*, plus other 3 nodes in the same *shard*, plus 3 nodes in the same *knot*, plus 3 nodes in the same *clot*, picking them from the routing table for FL0, FL1, FL2 and FL3 respectively.

- **FL5**; FL6 is inactive; Network size is between 1.048.576 and 33.554.432 nodes.

clots are organized in numbered <u>*clusters*</u> from 0 to 32. The number of active *clusters* is found in the consensus variable *L*.

Knowing that V[15] = 78, assuming *L* = 12 and following the same reasoning, coordinates for our example node would be:

{*cluster*, *clot*, *knot*, *shard*, *ring*}={2, 1, 4, 16, 4}

- **FL6**; Network size is between 33.554.432 and 1.07 billion nodes.

clusters are organized in numbered <u>*banks*</u> from 0 to 32. The number of active *banks* is found in the consensus variable *B*.

Knowing that V[14] = 242, assuming *B* = 3 and following the same reasoning, coordinates for our example node would be:

{bank, *cluster*, *clot*, *knot*, *shard*, *ring*}={0, 2, 1, 4, 16, 4}

Similarly, further *fractal levels* can be defined in a similar way in order to welcome and allocate more nodes. FL7 would enable a capacity of 34.36 billion nodes.

Summarizing the organizational levels, these are **node**, **ring**, **shard**, **knot**, **clot**, **cluster**, and **bank**. Each one contains 32 instances of the previous.

In a scenario where all 7 levels are utilized, the address space, which contains 2^160 addresses, is partitioned into approximately 33.5 million (32^5) partitions. Each partition is managed by its corresponding shard. An address is a 160-bit number, which can be represented by a decimal number with up to 48 digits.

12 Routing

Routing tables are a collection of references to other nodes that serve the purpose of establishing new connections. These tables are classified based on their fractal level (FL).

In this design, there are up to seven routing tables corresponding to FL0 to FL6. Each node maintains these tables individually and updates them on each peer-to-peer connection in a best-effort knowledge-sharing basis.

Every node would have a limited number of entries in each routing table, which are used by the clique manager to establish new connections for two primary purposes:

- routing protocol datagrams to the correct *shard*.
- Sharing *evidences* and performing consensus rounds within a *shard* and its *rings*.

RTFL0

The routing table for *fractal level 0* is a fixed size list with 31 entries. Each entry contains:

- Node id (public key hash).
- IP Address.
- TCP port (default port is 16672).

It is used by *clique manager* to connect with at least 3 random nodes for its FL0 connections.

RTFL1

The routing table for *fractal level 1* is a fixed size list containing 31 entries:

- *ring*. Universal unique id (uuid) identifying other blockchains in the *shard*.
- A fixed size list with 32 entries:
  - Node id.
  - IP Address.
  - TCP port.

Nodes working on the same blockchain as the node holding this routing table are excluded, since they are already in RTFL0.

Routing tables FL0 and FL1 are part of the *shard consensus*.

RTFL2 – RTFL6

These tables contain structures similar to RTFL1.

Routing tables are constructed through ongoing information exchange during node connections, and are designed to remain at reasonable sizes through algorithmic management. While they contain a generous number of entries, they are not expected to store all possible entries, as doing so would not be scalable in terms of storage capacity.

13 Clique manager.

The clique manager (CM) [29] is a crucial component responsible for maintaining stable connections with other nodes. These connections can last for hours or even days and serve to build a solid network for routing, evidence relay, and consensus.

Periodic changes to connections are introduced at random intervals through a process called mutation. This helps to randomize the participants and reduce the probability of collusion, which is a malicious activity performed by rogue nodes that can disrupt consensus.

Collusion is a harmful node behavior that can be carried out by malicious actors who modify the protocol with the intention of compromising the network (adversaries).

The CM would terminate current connections and establish new ones by randomly selecting connection setup parameters such as IP address and TCP port from the entries in the routing tables.

CM maintains at least 3 stable connections for each *fractal level*.

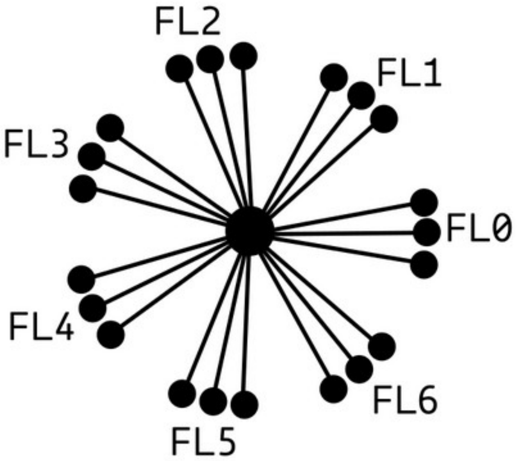


Figure 24: The clique manager is responsible for maintaining stable connections with other nodes for each of the 7 fractal levels. This means a minimum of 21 stable connections are kept in a configuration where 7 FL are active.

14 Storage of digital assets.

RIPEMD160 hashing can be used to reduce any piece of information into a fixed-length 20-byte vector of bytes, which is known as the asset ID.

In the "Encoded coordinates" section, we learned how to derive coordinates from node addresses. The same process can also be applied to any digital asset by first converting it into a 20-byte vector using RIPEMD160 hashing, which can then be used as the asset ID.

Coordinates {*bank*, *cluster*, *clot*, *knot*, *shard*, *ring*}can then be obtained for any asset-id.

Since all rings in a shard are designed to provide redundancy, the last coordinate "ring" can be omitted when obtaining coordinates {bank, cluster, clot, knot, shard} for any asset from its hash-id. Hence, the asset is stored in all 32 *rings* once the asset reach a node matching the *shard* and propagated.

When a new asset arrives, a node needs to decide whether to store it or forward it to other nodes. This decision is made based on a coordinate-matching ruleset.

The clique manager manages two modes of message forwarding:

- Stable connections
- Temporary short-lasting connections.

The clique manager attempts to optimize resource usage by maintaining three stable connections to nodes for each fractal level out of a possible 32. However, there may be situations where an asset needs to be forwarded to a coordinate that is not covered by the stable set. In such cases, the CM has several options:

- Create a new stable connection, potentially dropping an existing one, depending on available computational resources.
- Create a temporary connection to the correct node using the corresponding entry in the routing tables, forward the message, and then drop the connection.
- Forward the message to a node using any stable connection, even if it does not match the target coordinate.

The clique manager (CM) decides the most suitable action based on heuristics related to available computing resources. Apart from the stable connections, CM also has a rotating buffer of temporary connections that can be utilized on demand.

In the worst-case scenario where the asset doesn't match the bank coordinate, the asset would be forwarded to the cluster coordinate using one of the three connections in FL0. From there, it would be forwarded to the clot coordinate using one of the three connections in FL1, and then to the knot coordinate using one of the three connections in FL2. Finally, it would be forwarded to the shard coordinate using one of the three connections in FL3. Therefore, the asset would reach the destination shard in 4 hops.

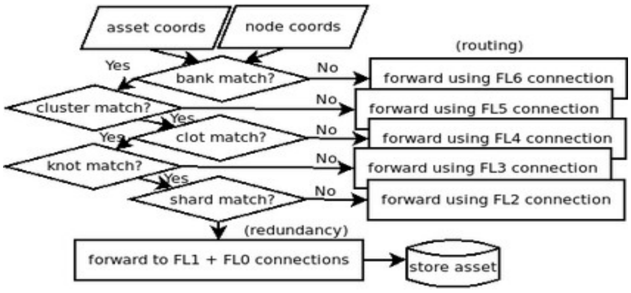

asset coords
node coords
(routing)
Yes
bank match?
No
forward using FL6 connection
cluster match?
No
forward using FL5 connection
Yes
Yes
clot match?
No
forward using FL4 connection
knot match?
No
forward using FL3 connection
Yes
shard match?
No
forward using FL2 connection
(redundancy)
forward to FL1 + FL0 connections
store asset

Figure 25: Flow diagram illustrating the decision making process of either forwarding the asset to the appropriate nodes or broadcasting it among the nodes in the destination shard.

Once the asset has reached the destination shard, it will be forwarded to all the remaining 31 rings in that shard. This way, all the 1,024 storage nodes in the shard would receive the asset using a cascade broadcasting algorithm, with the help of the FL1 and FL0 connections.

Once a node is connected to the shard with coordinates deduced by the asset-id, it can query the asset content using an API call. This API call would be sent to the node that stores the asset, which can then retrieve the asset content and send it back to the querying node.

There are certain types of assets in the system, such as *evidences* and *public algorithms*, that require special attention.

15 Public Algorithms.

Public algorithms (PA) serve as the Plebble equivalent to Ethereum's smart contract concept, as introduced by Ethereum [32]. These are programs that operate based on events and can save and load state from a chunk of bits or blob.

PA is submitted to the network as a special asset, with its id corresponding to the hash of the asset content, which is a compressed file.

Once the file is decompressed, a file tree containing source code files written in any general-purpose language and organized arbitrarily by the developer is revealed. This tree contains a makefile that implements standard targets to be invoked by the node to generate the corresponding executable binaries and store them for future use.

To implement PA, the Plebble SDK must be used, which enforces the implementation of certain interface functions, such as the evidence factory, evidence processing, and load and save state. The asset and its state blob are stored in an address corresponding to the asset-id, i.e., the PA address.

PA can be programmed using any general-purpose backend-oriented language that is machine code efficient, such as C, C++, or Rust. Although interpreted languages like Java, JavaScript, Python, and Bash are possible, they are discouraged for use with PA due to their higher demand for system resources than is strictly required.

PA executables are executed in a secure execution environment or "jail" and are never shared among other nodes. Only the source code asset is shared.

Plebble leverages the capabilities and compartmentalization (c18n) offered by CPU architectures that implement the CHERI architecture [6], such as ARM aarch64c.

PA publishes its own specifications for evidences, allowing users to create and broadcast instances of them. These instances are processed to produce state changes in the PA program.

There are two types of PA: user PA (UPA) and built-in PA (BIPA). The main difference between them is that UPA is executed exclusively by nodes belonging to its corresponding shard, while BIPA is executed by the entire network.

BIPA is considered a mass-consumption utility, while UPA has a limited audience. It is possible that once a UPA has proven its mass-consumption potential, it can be promoted to a BIPA. However, this process would require a software update.

Currently, the Plebble reference implementation includes four BIPA:

Cash. It manages the banking system's accounting for accounts and coin balances.

DFS. It manages the storage of digital assets in a distributed file system.

DNS. It handles name-resolution for stable wallet endpoints (QR) that allow wallets to change their IP address without disrupting service.

Sys. It manages software updates for the network.

16 Evidences

Evidences are a special type of asset because they are responsible for any change in the state of their corresponding publishing PA. Evidences published by UPA are routed to and processed by nodes in the same shard where the UPA is installed. On the other hand, evidences published by BIPA are routed to and processed by nodes in the shard/s corresponding to the addresses they refer to.

Evidence relay.

When new valid evidences are known by any node, they are propagated to their neighbors using a flooding algorithm, also known as a cascade or gossip mechanism in literature. The nature of this algorithm is similar to a breadth-first search algorithm in a graph, where propagation continues only if the evidence is valid and has not been seen before.

Evidence processing.

Once an evidence arrives at a node running its matching PA, it is processed by the PA. After validation, the evidence is positioned in the timeline/s associated with the addresses/s referenced in the evidence (BIPA) or the PA address (UPA). Each address has its own evidence timeline containing all evidences that contribute to the evolution of the address state, ordered by execution time. The evidences also form a linked list or chain, with each evidence referencing the previous one except for the first one.

The final state of a UPA or an address is the result of applying, in order, all available evidences that affect them. The sequence of evidences responsible for a final state of an address can be seen as an evidence chain, which is similar to a blockchain. However, a blockchain is designed to contain all possible addresses, which does not scale well. The evidence chain presented here only contains evidences affecting one address and is completely unaware of the state of any other addresses.

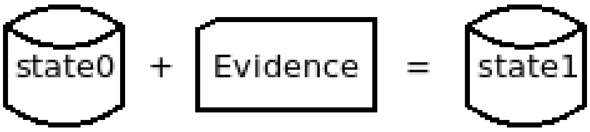


Figure 26: The next state of an address is the result of applying an evidence to the previous state.

17 Transactions

Evidences published and processed by the cash BIPA are commonly referred to as transactions (TX) for the purpose of comparison with Bitcoin.

Locking programs, unlocking inputs.

Each address contains a locking program, which is executed before any attempt to change the address state in response to evidences. By default, the locking program is a signature verification function. However, this can be replaced by any function defined by the user, such as multisig verification or a pay-to-redeem script.

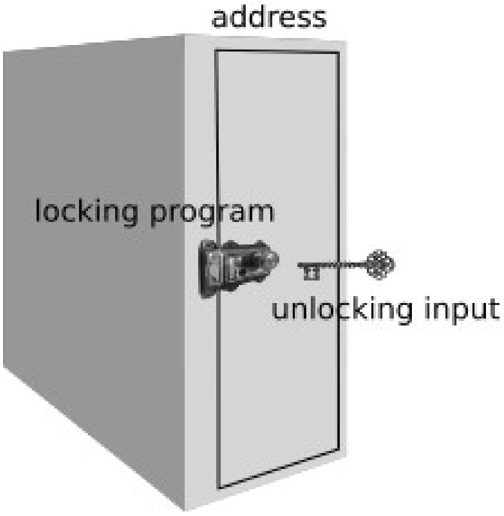


Figure 27: Addresses can be thought of as safe-deposit boxes, with locking programs serving as the open/close mechanisms. Just like a safe-deposit box needs a key or combination to be unlocked, an address requires unlocking inputs found in transactions in order to perform any changes to the state inside.

Outline of a transaction.

A transaction (TX) in the context of the Plebble architecture is composed of one or more sections, each of which refers to a particular coin or asset being transacted. Each section contains a list of inputs and a list of outputs. Inputs are addresses that represent the source of the assets being transacted, while outputs are addresses that represent their destination. In other words, a section of a TX can be thought of as a transfer of a particular asset from one or more source addresses to one or more destination addresses.

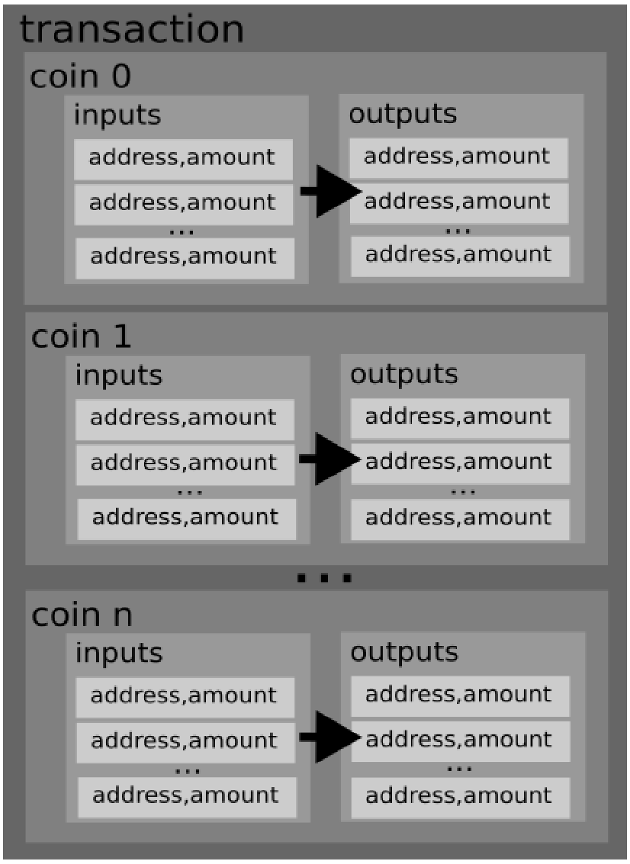


Figure 28: Outline of a plebble transaction.

Only the inputs in a transaction contain unlocking inputs that are needed to validate the transaction. The unlocking inputs are used to provide the necessary information to the locking program of each input address, allowing it to verify that the transaction is authorized to spend the coins associated with that address.

The outputs in a transaction don't need unlocking inputs, as they are simply defining new destinations for the coins being transferred. However, each output does include a reference to the address it is associated with, and that address's current state is updated with the new output as a result of the transaction.

Additionally, each output includes an identifier for the evidence that created it, allowing the history of the coins being transferred to be traced back through the chain of evidences affecting the address.

The sum of the inputs should be equal to the sum of the outputs, otherwise it will be rejected by the network. This is because the network wants to ensure that no new coins are created out of thin air or that coins are not destroyed. It ensures the validity of the transaction and the consistency of the ledger.

In the Plebble network, the fees are not decided by the users. Instead, the consensus algorithm automatically calculates the fees based on the complexity of the transaction and the current network load. The fees are then distributed among the nodes that process the transaction. This approach helps to ensure that the fees are fair and that the network remains stable and efficient even during times of high demand.

Transactions include a sigcode. To clarify the language, the sigcode in transactions is actually called the "signature code" and it allows users to specify which combination of inputs and outputs should be signed. This allows for flexibility in how transactions are constructed, including the ability to partially sign and send incomplete transactions to other parties for completion. This process can be repeated until the transaction is complete and ready to be broadcast to the network.

18 Consensus

Within the shard, as part of the consensus protocol, every node:

- Agree which other nodes are working on which blockchain replica (*ring*).
- Enforce IP address uniqueness for every participant in the *shard*.
- Enforce node onboarding rules, including a waiting list (*hall*).
- Agree on the pace of admission of new nodes (as part of evil/honest ratio control).

Such conditions are designed to observe an average proportion 80% honest / 20% evil.

This ratio comes from applying the *Pareto principle* to roughly determine the proportion of evil people in the world. The system shall be resilient under such condition.

In order to build a system from zero with a honest/evil proportion of 80/20 at a controlled pace, e.g. 1 node per minute, we fill a *big* waiting *list* of candidates (*hall* priming) and then onboard nodes randomly picking from it at the chosen pace.

The design leverages the **scarcity** and **cost** associated with *IPv4* addresses for sybil control.

19 Wallet

The term "wallet" refers to multiple components within the design that need to be distinguished based on context:

- The wallet subsystem includes all aspects related to key management, privacy, and trade automation.
- A user process is responsible for maintaining the user's privacy, safety, and interoperability.
- An online interactive agent that serves as a personal hub for dealing with society.
- The wallet daemon backend process containing algorithms for crafting *evidences*, e.g. signing monetary transactions, and safeguarding user secrets, e.g. *private keys*.
- The wallet front-end process.

There are two processes that represent the runtime wallet subsystem: a daemon and RPC clients.

For developers, Software Development Kits are provided for extending the different subsystems.

20 Wallet – SDK

Front End

The plebble-wallet Software Development Kit (SDK) is available in various programming languages to support the creation of new application front-ends that connect to the wallet daemon (back-end). The wallet SDK for each language provides an implementation of the wallet API and allows for easy extensibility, including synchronous and asynchronous function calls, handlers, and custom data transfer objects (DTOs) for passing function arguments and collecting responses.

To achieve the most efficient network communications with the SDK, datagrams over TCP/IP are used. However, remote access to the backend can also be enabled through a REST API interface on nginx or apache web server. While this option may incur a slight overhead caused by the REST specification, which utilizes the otherwise unnecessary HTTP protocol, it provides an alternative means of remote access.

R2R

The R2R-SDK is a distinct SDK used for creating wallet extension plugins that introduce specialized trading behavior to the back-end. These plugins define sub-APIs that assign specific roles to both parties involved in a peer-to-peer trading interaction. For example, roles such as doctor/patient, buyer/seller, player/casino, sports club/fan, CEO-CEO, CEO-CTO, etc. can be defined through the use of this SDK.

Each role in an R2R protocol is managed by a distinct plugin running on each side, allowing for personalized automation of each trade. These trades may be short-lived or long-term, and they can accommodate online-offline changes made by any of the participating peers.

21 Wallet – Functions

The front-end wallet API provides users with the following function categories:

- Balance query.
- Address and keys maintenance.
- Personal coin management.
- Cash transfers.
- Invoicing (half-baked *transactions*)
- Crafting, signing and relaying *evidences*.
    - Cash
    - On-Chain/Off-Chain Data Storage (Files, key-Value and time-series)
    - Software updates.
- Certificates maintenance (Graph of trust).
- Digest (hash) messages or files.
- Encryp/decrypt/encode/decode messages or files.
- Pairing RPC clients.
    - QR based connections.
    - Pre-pairing with PIN.
    - On-demand guest wallet creation.
- Manage guest custodial wallets.
- Create QR codes.
- Daemon control and monitoring.
- Peer connection status and checks dashboard.
- versioning info.
- Trading. Role-to-role protocols (R2R).
    - Business setup.
    - R2R plugins maintenance.
    - Business QR Codes that start trades straight away.
    - Bookmarks maintenance.
    - Access to specialized R2R sub-APIs.

22 Wallet – Daemon

The wallet daemon is a background operating system process that is responsible for safeguarding the user's private data, signing evidence, and communicating with other peers. This involves not only storing confidential information like private keys but also any other data that may be required to facilitate trades on behalf of the user's interests. This may include things such as Electronic Health Records (EHRs), pictures, documents, chats, and more.

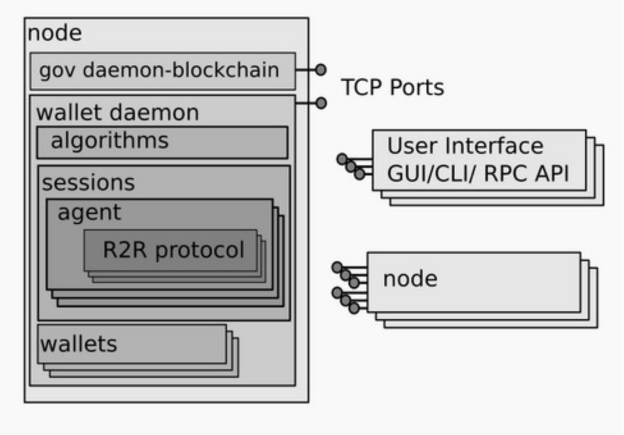


Figure 29: Components of a plebble node showing gov and (detailed) wallet daemon processes. The default TCP port for gov (public protocol) is 16672 and for wallet (private protocol) is 16673

Wallet address.

Each wallet is associated with an address that is derived from its private key, which is only known to the user. This address is generated upon the wallet daemon's initial launch and stored in the file *~/.plebble/wallet/k*. The address serves as a public identifier that enables wallets to connect and engage in trading.

Custodial wallet addresses.

Users who operate as daemon system administrators under the gov linux account, also known as sysops, have the option to configure their wallet to host guest wallets, which are also referred to as custodial wallets. These wallets are allocated for other users and are identified with an arbitrary string called *subhome*.

```
1.- 9210 wq9XTruXWKjAu3k55Ch8TJ8Qtsm
2.- wq9XTruXWKjAu3k55Ch8TJ8Qtsm.bob
3.- wq9XTruXWKjAu3k55Ch8TJ8Qtsm.bob TVcbPSgR6wueWS0
```

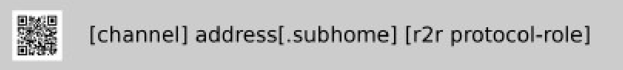
Drawing 1: Examples of a connection strings. 1.- A non-custodial wallet in a priavte network on channel 9210. 2.- A hosted custodial wallet ‘bob’ on channel 0. 3.- A shoping protocol with role seller run by bob’s custodial wallet.

Connection string. Service QR code.

To enable automation and conduct specialized business, users should be able to easily install and configure the corresponding R2R plugin in their wallet and generate a QR code that other users can scan to initiate a peer-2-peer trade between the two wallet daemons involved.

The connection string encoded in the QR is formed by the following fields:

- *channel*: This field contains a network identifier and is required if the wallets are on a different network other than channel 0. If this field is missing, it is assumed to be channel 0.
- *address*: This field identifies the remote wallet daemon.
- Subhome. If this field is missing, the connection is handled by the non-custodial wallet (root wallet). If present, the peer wallet is an inner guest wallet managed by the root wallet.
- R2R protocol-role: If this field is missing, both wallets will start a dialog that supports general features such as chat and sending files. If specified, both sides will start the corresponding R2R protocol with specialized automation for activities such as banking or shopping.



[channel] address[.subhome] [r2r protocol-role]

Figure 30: General pattern of a connection string used for starting trades with remote wallets.

It's worth noting that the Plebble blockchain integrates an Address Resolution System (ARS), which can be used to resolve wallet addresses to TCP/IP addresses. This may offer a more secure and decentralized alternative to traditional DNS systems, which are vulnerable to various forms of attack. By leveraging the Plebble blockchain's built-in ARS, users can potentially improve the reliability and security of their wallet address resolution process.

tsockets

The tsockets system library is an extension of the sockets library that adds an IPC (inter-process communication) layer, which provides a trading-session abstraction. This enables programs to use a context-rich environment for conducting trades, whether they are short or long-term, even in cases of network disconnections, whether intended or not.

When a user wants to initiate a trade with another wallet daemon, they supply a connection string to open a tsocket. The remote daemon, known as the follower, will also open a tsocket and attach the requested automation. This enables both sides to communicate with each other and conduct trades even if there are network disconnections or other issues. The tsocket system library provides a convenient way for programs to use this abstraction layer and handle trading sessions with ease.

The program conducting the trade, the R2R protocol, is given contextual information in a key-value structure after a successful connection. This information includes:

- unique trade id (tid, 20 bytes), common at both ends.
- Trade creation, last activity timestamps.
- Connection status: offline, online.
- Trade status. State machine.
- Remote endpoint info: wallet id, IP:Port
- Remote verified *personalities* (verified identity)
- Peer profile based on *personality*. Credit score.
- Local *personality* in use.
- Activity Log file and local storage path.
- Remote and Local available *R2R protocols* that can be started.
- Specific information from current ongoing *R2R protocols*.
- Workflows synchronization information for shared documents management.

Trades conducted through the R2R protocol are persistent and their context and history are preserved even if the underlying socket is disconnected. This means that a trade can be stopped at any point and resumed later without losing any information or progress made. Additionally, human actions can be inserted at any point within the automation, allowing for manual intervention if necessary. This persistence and flexibility make the R2R protocol well-suited for conducting complex, long-term trades with multiple parties.

Personalities, also known as anonymized digital identities, play an important role in the R2R protocol as they allow for the creation of persistent identities that are not linked to the user's real-world identity. This enables users to conduct trades and interact with others in a pseudonymous manner, protecting their privacy while still building trust based on their past interactions.

Personalities are created and managed by the wallet daemon and are used to generate unique trade IDs (tids) for each trade, which are shared between both parties in the trade. The use of tids allows for the tracking of trade history and context even in the case of disconnections or resumptions.

Personalities also include information such as credit score, which can be used to evaluate the trustworthiness of a peer and make informed trade decisions. The use of personalities and their associated profiles is an important aspect of the R2R protocol in building trust and confidence in the trading process.

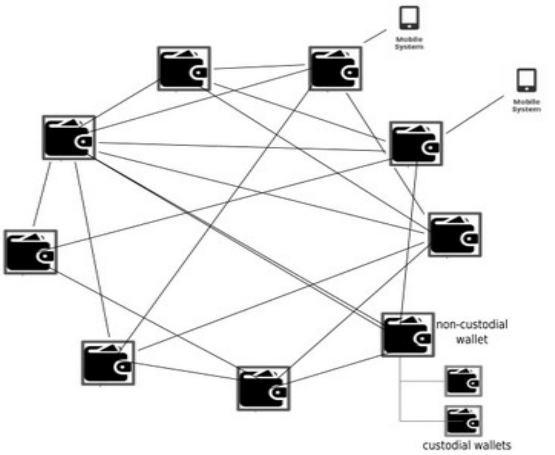


Figure 31: Wallets controlled with cellphones doing P2P trades between them. The difference between a custodial wallet and a non-custodial wallet is the former run in other's computer (cloud scheme).

23 Role-2-Role (R2R) protocols

R2R protocols are used to facilitate P2P interactions within a trading context, and are powered by separate programs known as wallet plugins. These plugins are responsible for executing the automation necessary to conduct the trade.

The plugin architecture in the wallet system is supported in both the backend and frontend, which enables developers to create custom interactions between actors of a Distributed Autonomous Organization (DAO) using the wallet-r2r-SDK. This allows for a more flexible and customizable system, as developers can create their own specialized features and protocols within the wallet system.

There are two classes of R2R protocols - symmetric and asymmetric. Symmetric protocols involve both parties adopting the same role, such as two banks conducting a transaction. Asymmetric protocols involve two parties adopting different roles, such as a buyer and seller or a patient and doctor in a healthcare context.

Developing a R2R protocol involves the specialization of several base objects provided in the SDK. These objects include:

- The "**business**" class: This is a singleton class representing the user's central control over their business. It manages databases, coins, imagery, marketing, accounting, and other common information and algorithms that are not related to a specific interaction but to all of them. It serves as a factory for instantiating protocols for each individual connection with peers.
- The "**protocol**" class: This class is instantiated for every trade with a specific peer when required. It controls the interaction and is where the automation logic resides. A tailored subprotocol can be easily defined with the apitool and implemented as an extension of the base protocol. For example, the bank role defines an API with functions for transferring coins, invoicing, etc.
- The "**workflows**" class: This class consists of document definitions (compatible with the W3C Verifiable Credentials standard) and flow definitions, or traveling directions. Workflows facilitate the construction of bureaucracies within a DAO that can be leveraged by higher decision-making algorithms embedded in the protocol class.
- Front-end **fragments**: UI/UX classes in different languages are included for two main classes of Human-Machine Interfaces (HMI). These include a text console, which includes the definition of menus, help screens, and shell input handlers, and graphical user interfaces (GUIs), which provide flexibility for developing user dashboards that will be instantiated by the GUI wallet front-end on user request. This gives users display and control over the different instances of the protocol.

For symmetric protocols, the same set of classes can be used for both parties, while for asymmetric protocols, separate sets of classes must be defined for each party, reflecting their distinct roles in the trade.

The R2R protocol-specific software is packaged as shared libraries, with typically one shared library per role, and optionally a common library with shared functionality. Additionally, one library per workflow specification can be created, which once deployed, are installed in the plugins directory of the wallet. The default directory for these shared libraries is often something like *~/home/gov/.plebble/wallet/trader/lib*.

The distribution of R2R packages can be done by the author under any chosen license, whether it is free or proprietary. For free software, the source-code form can optionally be distributed, and the final user can perform the compilation. Alternatively, binary packages can be distributed for closed-source software.

R2R plugins run attached to the wallet daemon process, which allows the front-ends to access the plugin's functionality seamlessly. This also enables the front-ends to request and load specific fragments on demand, providing an extensible and flexible user experience with a range of features.

To clarify, the wallet daemon manages both the root wallet (which is typically non-custodial) and guest wallets (which can be either custodial or non-custodial). R2R protocols can be enabled and disabled for individual wallets, regardless of whether they are custodial or non-custodial. This allows for flexibility in how users interact with the protocols and manage their funds.

Built-in R2R Protocols.

The Plebble distribution currently includes several R2R protocols:

- **Bank**: a symmetric protocol that offers banking services such as money transfer, loans, and a transaction obfuscation service (mixer).
- **Shop**: an asymmetric protocol designed for online shopping experiences.
- **Curex**: an asymmetric protocol that enables users to place cryptocurrency and fiat currencies buy/sell orders. It includes a matching engine that uses fiat payments gateways like stripe, and allows users to operate their own exchange.

24 Zero-Knowledge Proofs

While Zero-Knowledge Proofs (ZKPs) and STARKS are both cryptographic techniques for privacy and security, they are not the same thing. ZKPs are a type of proof that allows one party to prove to another party that a statement is true, without revealing any other information beyond the truth of the statement. ZK-SNARKS is a specific type of ZKP that stands for Zero-Knowledge Succinct Non-Interactive Argument of Knowledge, which is used in blockchain technology for privacy and scalability.

On the other hand, STARKS (Scalable Transparent ARguments of Knowledge) is a different type of proof system that also allows for verifiable computation without revealing any private information. STARKS is an evolution of ZK-SNARKS, which is more transparent, scalable, and requires less trust in the setup phase.

That being said, integrating ZKPs and/or STARKS into wallets can improve the privacy and security of transactions by allowing users to prove that they possess certain information or assets without revealing additional details. This can help prevent fraudulent activities and protect user privacy.

By incorporating this cryptographic technology, a new type of privacy-preserving interactions can be achieved, enabling users to provide cryptographic proofs that can be trusted without exposing their private information. This can help establish trust and confidence during the course of a trade.

Consider a scenario where Alice and Bob are engaged in a trade, and Alice needs to verify if Bob worked for the company TimeX between specific dates. Bob possesses a certificate containing the required information, along with other private data like his salary. However, he is not willing to disclose his salary at this stage of the trade. In a traditional setup, Bob would have to reveal the entire document to Alice for her to perform signature verification and extract the relevant information. However, this would give Alice an unfair advantage by allowing her to access Bob's private information. Since the complete document is required for signature verification, Bob cannot selectively reveal only the required information.

Using the emerging ZK-STARK technology, the following process would be carried out instead: Alice creates a program that takes a signed document as input, verifies the signature, reads the dates, and outputs either PASS or FAIL based on the values found. She sends the program to Bob for review to ensure it doesn't reveal any unwanted information, and Bob then compiles and runs it against his signed certificate from TimeX using the plebble-ZK processor on his computer. Upon execution, the program generates an output and a proof, which Bob sends back to Alice. Using the same infrastructure, Alice can verify the output using the proof and be assured that Bob didn't modify the program or alter the outputs. Alice now knows that Bob worked for TimeX during the requested period, but she cannot learn any other details, such as the actual start/end dates or Bob's salary. Importantly, the certificate never leaves Bob's computer, and no private information is disclosed during the process.

When a predefined generic infrastructure is in place, the process can be simplified for the user. This involves using standardized certificates that contain key-value fields and predefined field-based proofs. With these templates, users do not need to exchange source-code or verify it, as they can rely on built-in defaults. This makes the process as simple as specifying conditions based on field values, with the remaining interactive process fully automated until the result is ready to be consumed by the proof issuer.

The emergence of Zero-Knowledge proofs represents a significant shift in the way we can establish trust without compromising privacy, making it a disruptive technology and a game changer in the field.

25 Governance

Our goal is to promote a high level of distribution in blockchain governance through a unique approach that involves independent release authorities, brands, and their respective communities.

development communities

One of the main weaknesses of projects that produce distributed systems is the centralization of development. Although they often rely on open development communities, the work is typically funneled through a centralized authority that controls the release process of the node software.

Our objective is to enhance distribution in blockchain governance by reducing the reliance on the traditional centralized scheme. Instead, we aim to implement a more redundant system where multiple authorities are involved, each with the power to push updates to only a limited number of nodes.

Software updates

The original source code for the node software is held by the initial author or the genesis node. The code is then compiled and transformed into a distribution file called an upgrade-blob, which contains binaries for supported CPU architectures, as well as sources, resources, and an install script called install_blob. When the install_blob is executed, the upgrade sequence is performed and the node software is restarted.

Once the blob is produced, it is made public by being uploaded into a blockchain account owned by the author. In the case of Plebble, updates are published in the address 4NwEEwnQbnwB7p8yCBNkx9uj71ru. The genesis node serves as the first distribution node, also known as the brand genesis node or brand distribution end (BDE).

New users can download the software with the help of an existing node that publishes both the blob and the install script. All Plebble nodes synchronize from the aforementioned address.

Users have the option to voluntarily mutate their node and become a BDE for a new brand by using a different address to distribute the software. They can use the Plebble source code as upstream input, and optionally modify it with their own variations and branding. They can then build new blobs and upload them to the blockchain for other nodes to synchronize with.

This process effectively removes centralization in the development of the system by distributing the software production across an unlimited number of BDEs. This creates a decentralized and collaborative environment where users have the ability to modify and improve the software, and share their changes with the community.

There is a potential risk for a BDE to become rogue and produce software that is unable to cooperate with other BDE nodes or introduce malware. However, this problem can be mitigated through a well-distributed BDE ecosystem. According to the Pareto rule, assuming 80% of BDEs are honest and 20% are malicious, the network can still be resilient. This assumption is better aligned with reality with a large number of BDEs.

The consequences of malicious behavior are constrained by the number of downstream nodes that the malicious BDE is able to spawn, which is assumed to be much less than the number of downstream nodes that honest BDEs are able to spawn. As a result, the network is able to maintain its integrity and security despite the presence of malicious actors.

As previously mentioned, BDEs are responsible for building nodes and transforming source code into blobs. The number of BDEs in the network is a critical factor in determining how decentralized the software distribution activity is. A larger number of BDEs results in a more decentralized network where the distribution of software is more evenly spread out, reducing the risk of centralization and increasing the resilience of the system.

As release authorities, BDEs have full control over the quality of the source code they use as input for the release. The source code lifecycle is a crucial factor in maintaining a healthy distributed governance model, where no single person or organization has complete control over the software being run by nodes worldwide. By having control over the source code, BDEs can ensure that the software being released is of high quality and meets the standards of the community. This decentralized approach to governance promotes transparency and accountability, ensuring that the interests of all users are taken into consideration.

BDEs typically have their own development communities, and they collect improvements from their communities using various channels such as BDE accounts on social media sites like GitHub or GitLab, or even through naked Git. These changes are selectively propagated upstream the BDE graph, giving the parent BDE (i.e., the one the BDE forked from) the opportunity to integrate changes coming from both upstream and downstream.

As release authorities for their brand, BDEs have the discretion to merge interesting features or rule out non-interesting ones according to their criteria. This allows for a collaborative approach to software development, where the best ideas and improvements are integrated into the software while avoiding unnecessary or conflicting changes. By working in this decentralized and collaborative manner, the software is able to evolve in a way that best meets the needs of the community, while maintaining the integrity and security of the system.

The decentralized and collaborative approach to software development through BDEs allows for distributed governance, which is a unique feature in the web3 space. The scheme enables local authorities (i.e., BDEs) to make decisions about which features are pushed to nodes feeding from them, rather than having a system-wide guardian role that blocks suggested features.

This approach gives developers the freedom to choose from a wide range of BDEs to push changes to, or to become BDEs themselves. When developers run their own BDEs, their changes are pushed to nodes connected to them and are also propagated to other BDEs. The other BDEs then have the discretion to accept or reject these changes for their brand. This allows for a more democratic and flexible system of software development, where individual BDEs have more autonomy to make decisions that best serve their community's needs.

26 Plebos Operating System

Node software offers a comprehensive operating system that is ideal for use in routers, virtual machines, and affordable dedicated computers such as Raspberry Pi. This operating system can also be utilized on desktops, laptops, and tablets, and can be developed and distributed by BDEs.

Our proposed operating system is a fully-featured solution for both servers and desktops. It is based on the reliable Debian GNU/Linux, FreeBSD, or CheriBSD, operating at the kernel level. The system includes user wallets and supports public processes that utilize public algorithms, commonly referred to as smart contracts. These processes are displayed alongside regular (private) processes and can be managed using modified versions of regular tools such as ps and kill.

In addition, the operating system incorporates progressive consensus algorithms that utilize both progressive (or lazy) and finalization algorithms to reach a consensus on arbitrary data, such as a ledger. While finalization algorithms can't scale to billions of nodes and require specialized computers, progressive algorithms take a slower pace until consensus converges. For instance, it takes around 60 minutes (or 6 blocks) for a transaction to settle in the Bitcoin network using this approach.

Our objective is to establish a large-scale network where security is based on the sheer number of low-profile nodes, so their participation has a minimal or unnoticeable impact on the system resources usage.

We strongly believe that integrating core functionality is the most effective approach to propel technology towards new and innovative ways of utilizing internet infrastructure. Our focus is on privacy-preserving, serverless, peer-to-peer interactions, and our goal is to contribute to the development of Web3 technology.

27 Power consumption

Right from its inception, Plebble has been purposefully crafted to establish a network powered by non-expensive nodes. Unlike traditional distributed ledger architectures, Plebble operates without Proof of Work (PoW), which eliminates the need for mining.

Furthermore, the codebase, developed in the highly-efficient programming language C++, is optimized for speed and efficiency. Our testing has demonstrated that Plebble runs seamlessly on low-profile hardware, such as Raspberry PI Zero, without compromising on performance.

Figure 32: Raspberri Pi Zero is a low power computer. Via de micro USB interface it runs at 5.2V. When connected to WiFi it draws around 200mA which is a total of 1W.

28 Enhanced Security

While we have implemented a software stack that can be deemed secure and has demonstrated resilience against external attacks, it is crucial to bear in mind that it is never possible to provide complete assurance of security. As the saying goes, "software always has bugs," and unforeseen vulnerabilities can always arise.

Therefore, we understand the importance of remaining vigilant and continuously monitoring our system for any potential issues. By keeping this mantra at the forefront of our approach, we can proactively identify and address any security concerns that may arise, ensuring that our system remains as secure as possible.

Future glitch

A system that undergoes continuous evolution and development runs the risk of introducing regressions or new glitches that may pass initial testing but could eventually be shipped into production, potentially compromising the privacy of user data.

To mitigate this risk, we implement rigorous testing protocols, including comprehensive quality assurance and security testing, to identify and address any potential issues before they can be released into production. Additionally, we have established an active monitoring system to detect any anomalies or breaches that may occur after the system is launched.

Moreover, we value transparency and are committed to promptly addressing any issues that may arise. We work closely with our user community to report any discovered vulnerabilities or potential areas of concern, providing timely updates and implementing necessary measures to ensure the security and privacy of user data.

CHERI

The introduction of a new generation of CPU architecture called CHERI (Capability Hardware Enhanced RISC Instructions) has the potential to significantly enhance system security by reducing successful attacks. CHERI introduces an extended set of instructions that allow for temporal and spatial memory access protections to be enforced at the hardware level.

By incorporating CHERI into our system, we can provide an additional layer of protection against potential security threats. This hardware-based security approach can effectively reduce the risk of successful attacks by enforcing strong memory access protections. Additionally, this technology can help to address potential vulnerabilities in existing software applications that may be exploited by attackers.

Overall, we recognize the importance of continuously evaluating and integrating new technologies, such as CHERI, to enhance the security and resilience of our system against potential threats.

We have fully leveraged the capabilities defined in the CHERI specification [6], developed by the University of Cambridge, to enhance the security of our codebase. Our team has conducted extensive testing of our system using an experimental SoC, the Morello board [7], supplied by ARM. This testing was conducted as part of the Digital-Secure-by-Design [8] initiative, which is supported by UK Research&Innovation [9].

By integrating CHERI into our system and testing it using cutting-edge hardware like the Morello board, we have demonstrated our commitment to leveraging the latest technological advancements to enhance the security of our system. We believe that this approach will enable us to provide the highest level of security and resilience possible to our users.

29 Reference Implementation

We have taken great care to ensure that our codebase is well-structured, maintainable, and follows clean-code guidelines. Our reference implementation is split into logic layers, allowing for efficient development and maintenance.

For the backend, we have selected C++ as the language of choice, providing maximum performance and efficiency for running on small devices such as IoT devices, mobile devices, and embedded internet routers. We have extensively tested our backend on devices such as the Raspberry PI Zero.

To provide a user-friendly front-end, we utilize a variety of languages, including C++, Java, and WebAssembly (WASM). Our SDK is designed to be easily extensible to support additional programming languages.

We have also developed a low-footprint network protocol that runs over the ubiquitous TCP/UDP/IP stack. This protocol is designed to minimize overhead and eliminate the need for higher-level protocols and data representation formats such as HTTP, HTTPS, JSON, and XML.

Our source code is available on GitHub at [51], where we welcome contributions from the community. Additionally, we have published a landing page for Plebble at https://plebble.net, which includes links to our community resources.

Acknowledgements

To my family Aida, Axel and Alvaro, parents Marcos and Sole, Aida’s parents Pedro and Pilar for their love, patience, generosity, cooking skills and enthusiasm for whatever I do.

To my FFF group (Friends, Family and Fools) who helped bootstrapping the first nodes: Carlos, Queque, Helena, Javi. Felipe, Noemi, Noe, Ruben, Judith, Trucha, Koncio, Billy, Pupilo, Ferni, Mayte, Arancha, Nono, Julio Jorge, Tania, Mejuto, Fernando Norm, Carras, Tomas, Freddy, Paco Ferre, Cesar Fuentes, Maite ESA, Simeon, Rene and Manju.

I would like to thank Satoshi Nakamoto, for publishing the inspirational material that triggered the crypto movement, fueled my research and development on the generic goal of blueprinting the plumbing for a better world leveraging the full potential the ubiquitous internet has to offer in untested ways, within an environment of continuous breakthroughs, news, passion, hype and fun.

To those that opposed me in insane way making me stronger, whose names I try to forget.

The next *Oscar* goes to the Plebble community, small but supportive during development of the Plebble codebase.

I would like to thank Professors Simeon Nelson and Ljubomir Jankovic for opening a Doctorate position in the Department of Future Societies of the University of Hertforthshire, partially inpired from conversations with me about the blockchain.

Last but not least, I’d like to thank Mr. Edward Cole for funding and making this work possible, adding in addition a great deal of knowledge, passion and commitment. Also for founding and running KATLAS Technology, a Tech startup based on *Plebble* committed to bring *digital transformation* in the business space. He is also

responsible for coining the name *plebble*, a twist around '*us*' as plebs.

References

[1] Civilization types.

https://en.wikipedia.org/wiki/Kardashev_scale

[2] Standards for Efficient Cryptography.

http://www.secg.org/sec2-v2.pdf

[3] Bitcoin: A Peer-to-Peer Electronic Cash System. Satoshi Nakamoto.

https://plebble.net/bitcoin.pdf

[4] Verifiable credentials.

https://www.w3.org/TR/vc-data-model

[5] RIPEMD160.

https://homes.esat.kuleuven.be/~bosselae/ripemd160/pdf/AB-9601/AB-9601.pdf

[6] CHERI Architecture.

https://www.cl.cam.ac.uk/research/security/ctsrd/cheri

[7] ARM Morello.

https://www.arm.com/architecture/cpu/morello

[8] Digital Secure by Design.

https://www.dsbd.tech

[9] UKRI.

https://www.ukri.org

[10] One-Way HASH.

http://www.aspencrypt.com/crypto101_hash.html

[11] The Internet of Money Vol III – A.M. Antonopoulos. P37. ISBN 9781947910171

[12] Swarm Intelligence.

https://www.sciencedirect.com/topics/engineering/swarm-intelligence

[13] Tor. The Second-Generation Onion Router.

https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.pdf

[14] Proportional block reward as a price stabilization mechanism for peer-to-peer electronic cash system.

https://ergon.moe/prop-reward.pdf

[15] Bitcoin cash.

https://bitcoincash.org

[16] There is one global chain.

https://nakamotostudies.org/emails/satoshi-reply-to-mike-hearn

[17] Ethereum. https://ethereum.org/en/whitepaper

[18] IOTA.

https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf

[19] Namecoin.

https://www.namecoin.org/resources/whitepaper

[20] Litecoin.

https://litecoin.org

[21] Capitalization ranks.

https://coinmarketcap.com

[22] Nano.

https://www.exodus.com/assets/docs/nano-whitepaper.pdf

[23] The blockchain trilemma.

https://vitalik.ca/general/2021/04/07/sharding.html

[25] Debt: The first 5000 years. David Graeber. ISBN 978-1612194196

[26] The Blocksize War. Jonathan Bier. ISBN 9798721895609

[27] Datagram implementation.

https://github.com/root1m3/plebble/blob/main/core0/us/gov/socket/datagram.h

[28] Service API for gov and wallet.

https://github.com/root1m3/plebble/tree/main/core0/us/api/netsvc

[29] Clique manager.

https://github.com/root1m3/plebble/tree/main/core0/us/gov/peer

[30] DFS. Built-in Distributed File System.

https://github.com/root1m3/plebble/tree/main/core0/us/gov/dfs

[31] k-d tree.

https://en.wikipedia.org/wiki/K-d_tree

[32] Ethereum Smart contracts.

https://ethereum.org/en/smart-contracts

[33] The Cathedral & the bazaar. Eric. S. Raymond. Musings on Linux and Open Source by an accidental revolutionary.

[34] The Wisdom of Crowds. James Surowiecki. Why the many are smarter than the few.

[35] The Hitchhiker’s guide to the galaxy. Douglas Adams. Out-of-the-box thinking.

[36] Inventing the Future. Postcapitalism and a World Without Work. Nick Srnicek, Alex Williams. Demand full automation. Demand Universal basic income.

[37] Queremos su dinero. Jesus Martinez del Vas. The story of AMSTRAD Spain. Musings on consumer electronics distribution.

[38] The history of ARM on its 25th aniversary. ISBN 9781906615956

[39] Crypto. Steven Levy. ISBN 9780140244328

[40] Anarchy in the Organism. Simeon Nelson. ISBN 9781908966285

[41] Manufacturing Consent. Edward S. Herman & Noam Chomsky. ISBN 9780099533115 The political economy of the mass media.

[42] We are Anonymous. Parmy Olson. ISBN 9780434022083 Inside the hacker world of LulzSec, Anonymous and the Global cyber Insurgency.

[43] Just for Fun. Linus Torvalds. ISBN 1587990806 The story of an accidental revolutionary.

[44] The Snowden Files. Luke Harding. ISBN 9781783351046 The inside story of the world’s most wanted man.

[45] Doughnut Economics. Kate Raworth. ISBN 9781847941398 Seven ways to think like a 21st Century Economist

[46] The Infinite Machine. Camila Russo. ISBN 9780062886149 How an army of crypto-hackers is building the next internet with Ethereum.

[47] Crypto Economy. Aries Wanlin Wang. ISBN 9781510744820 How blockchain, cryptocurrency and token-economy are disrupting the financial world.

[48] Attack of the 50 foot blockchain. David Gerard. ISBN 9781974000067 Bitcoin and blockchains are not a technological story, but a psychology story.

[49] Scalability rules. Martin L. Abbott, Michael T. Fisher. ISBN 9780134431604 Principles for scaling web sites

[50] The mythical man-month. ISBN 0201006502 Essays on Software Engineering

[51] Plebble Reference Implementation.

https://github.com/root1m3/plebble

[52] KATLAS Technology Landing page.

https://katlastechnology.com

[53] apitool. API maintenance.

https://github.com/root1m3/plebble/tree/main/core0/us/api

[54] apitool input files.

https://github.com/root1m3/plebble/tree/main/core0/us/api/data

[55] apitool output. service numbers

https://github.com/root1m3/plebble/tree/main/core0/us/api/netsvc

[56] *plebble bank* R2R symmetric protocol.

https://github.com/root1m3/plebble/tree/main/core0/us/wallet/trader/r2r/w2w

[57] *plebble shop* R2R asymmetric protocol.

https://github.com/root1m3/plebble/tree/main/core1/us/trader/r2r/bid2ask

[58] M. Pwase, R, Shostak, L. Lamport - Reaching Agreement in the Presence of Faults.

http://lamport.azurewebsites.net/pubs/reaching.pdf

[59] Leslie Lamport - Time, Clocks, and the Ordering of Events in a Distributed System.

http://lamport.azurewebsites.net/pubs/time-clocks.pdf

[60] PBFT - Practical Byzantine Fault Tolerance.

https://pmg.csail.mit.edu/papers/osdi99.pdf

[61] SBFT - A Scalable and Decentralized Trust Infrastructure.

https://arxiv.org/pdf/1804.01626.pdf

[62] Pastry - Scalable, decentralized object location and routing for large-scale peer-to-peer systems.

https://rowstron.azurewebsites.net/PAST/pastry.pdf

[63] CAN - A Scalable Content-Addressable Network.

https://people.eecs.berkeley.edu/~sylvia/papers/cans.pdf

[64] Chord - A Scalable Peer-to-peer Lookup Service for Internet Applications.

https://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

[65] Tapestry - A Resilient Global-Scale Overlay for Service Deployment.

https://pdos.csail.mit.edu/~strib/docs/tapestry/tapestry_jsac03.pdf

[66] Kademlia - A peer-to-peer Information System Based on the XOR Metric.

https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[67] Our-System The quest for scaling BFT Consensus through Tree-Based Vote Aggregation.

https://arxiv.org/pdf/2103.12112.pdf