VeloxChain

CONNECTING URBAN MOBILITY ON BLOCKCHAIN

Veloxchain.io

Connecting Urban Mobility on Blockchain

Version 1.8

Authors:
Smart Urban Technologies Pte. Ltd d.b.a VeloxChain

1. Introduction 3
2. VeloxChain's Design Principles 3
3. VeloxChain Infrastructure Technologies 7
3.1. Privacy-Preserving Solution with Zero-Knowledge Proofs 7
3.2. Privacy-preserving collaboration in VeloxBiz 8
3.2.1. Anonymous Collaboration Layer with Zero Knowledge Proof 9
3.2.2. Private Keyword Search Engine 13
3.3. VeloxChain Consensus - Proof of Stake 15
4. VeloxChain DApp's Architecture 18
4.1.1. Meta Transaction Relay Protocol 20
4.1.2. Payment Channel 22
4.1.3. Vehicle Ownership and Tokenization from ERC721 22
5. Reference 23

1. Introduction

Ethereum[2] is a decentralized computing platform where users deploy smart contracts to perform computations. Nobody controls the Ethereum network and users completely control their own smart contracts - it is a trustless environment. This is a fantastic platform on which to build a decentralized ecosystem which uses smart contracts to process transactions.

One of the application areas for blockchain is Internet of Things (IoT). For example, BMW, General Motors, Ford and Renault[3] have launched their own blockchain research projects for the automotive industry, seeking to leverage IoT and blockchain technologies for a smart transport network.

VeloxChain is an open-source blockchain-based protocol and marketplace for shared mobility services. Our ambition is to democratise the sharing mobility market and accelerate the world's transition to an open and seamless mobility. We provide a one-stop solution for all participants in the ecosystem from developers, mobility providers and end-users, ultimately, benefiting from global network effects.

VeloxChain is empowered by several critical technologies:

1. Privacy-preserving solution by Zero-knowledge proofs.
2. Enterprise Collaboration Solutions leveraging Zero-Knowledge Proof and cryptographic search algorithms on VeloxBiz.
3. VeloxChain consensus Proof Of Stake.
4. Meta Transaction Relay Protocol
5. Vehicle asset tokenization developing from ERC721.

In the following sections we will explain our technologies in detail, beginning first with blockchain.

2. VeloxChain's Design Principles

Blockchain technology leads fuzzier APIs to allow more agile integration of new systems, automate data and payment process among stakeholders; however, blockchain technology is still at the early stage and in order to meet rigorous end-user applications requirements such as **low latencies, immediate transaction finality, high performance, excellent scalability and support multi-level of data privacy**.

Ethereum with less than 15 tx/s with 15 000 nodes[4] and delaying Ethereum's Casper (Proof of Stake) and Sharding until 2020[5], even Zilliqa blockchain with sharding solution only achieves 2488 tx/s with 3600 nodes or EOS achieves 3097 tx/s[6] with Delegated Proof-of-Stake and 21 block validators. Blockchain protocols are a form of distributed system, it inherits properties from CAP theorem - Consistency, Availability and Partition tolerance. Especially, the public protocol faces more difficulties where is unable to control the bandwidth and hardware limitation of random public miners (block validators). Each consensus has own advantages and disadvantages in terms of security. At the time of writing, **there is no existing one-size-fit-all protocol which achieve planetary-scale with million transactions to serve all applications in the world**;

Beyond scalability consensus solutions, there are two promising scalability directions:

- **Sharding**[11]: Similar to sharding concept in traditional database distributed system, shading divides the chain into small groups (shards) for parallel transaction processing, to increase transaction throughput of the whole blockchain ecosystem. There are two main stages in sharding solutions: transaction sharding and stage sharding. State sharding is more complicated and challenging as involving cross-shard communication where the receiver is a smart contract account. Zilliqa was the first protocol implementing sharding solution; however currently it is not able to store commitment of zero knowledge of proof. In our view point, sharding in blockchain still at the early stage.
- **Multi-chain**: this solution allows multi child chains sync and communicate with each other through a parent chain. For example, Plasma sidechains such as Loom network[12] with DPOS to interact with Ethereum; however, it's not practical with the current Ethereum transaction speed (15 tx/s). And

interoperability blockchain projects like ICON, Polkadot[14] and AELF are also potential solutions to improve decentralized features.

The major feature of blockchain is transparency which is demonstrated well by the first blockchain - BitCoin and Ethereum is the distributed and transparent computer which no one entity controls the network. Anyone can write or read data to the Ethereum network. Although Ethereum users addresses the privacy problem by issuing pseudonymous addresses (a pair of ECDSA public-private key), it is still possible to find out who's addresses they are thought various techniques. The practical solution is to use homomorphic encrypted end-to-end transactions mean only the parties involved in the transaction revealing the data. Homomorphic encryption (eg Zero-knowledge proofs) allows for computations to be done on encrypted data without first having to decrypt it. In other words, this technique allows the privacy of the data/ transaction to be preserved while computations are formed transparently on blockchain, without revealing that data/ transaction. However, there are two major challenges making homomorphic encryption becoming production in the blockchain industry.

- Firstly, a computation performed on the encrypted data usually is much higher than a normal transaction; it means computing (encrypt (A) + encrypt (B)) >> computing (A+B). Thus, **it is unrealistic to execute proof transactions directly on Ethereum blockchain** because Ethereum transaction is very costly.

- Secondly, as mentioned above, blockchain protocols are a form of distributed system; therefore, blockchain consensus is maybe familiar with senior engineers who work on high distributed database computing system like Google, Facebook or Uber. However, **implementing homomorphic encryption techniques required DApp developers having advanced cryptographic knowledge** to provide strong mathematically provable guarantees for the privacy of data and transaction.

VeloxChain addresses these issues by developing Proof-of-Stake chain with near-zero transaction fee while VeloxDev packages allow Dapp developers implement Zero-knowledge proof with miminum cryptographic knowledge.

An undeniable fact is that Ethereum has the largest developer community of decentralized applications with various development tools and Solidity is a standard learning smart contract language for DApp developers. Therefore, **VeloxChain provides the same functionalities as standard Ethereum to migrate Ethereum developers build DApps on VeloxChain without learning-curve**; this means that:

- VeloxChain uses Elliptic Curve Digital Signature Algorithm ECDSA)[7] to generate public-private keys and authenticate the signature. ECDSA uses the algebraic structure of elliptic curves over finite fields ($y^2 = x^3 + ax + b$). Similar with Ethereum, VeloxChain use a Koblitz curve secp256k1[8]:$y^2 = x^3 + 7$ (where a =0 and b = 7), it looks like this:

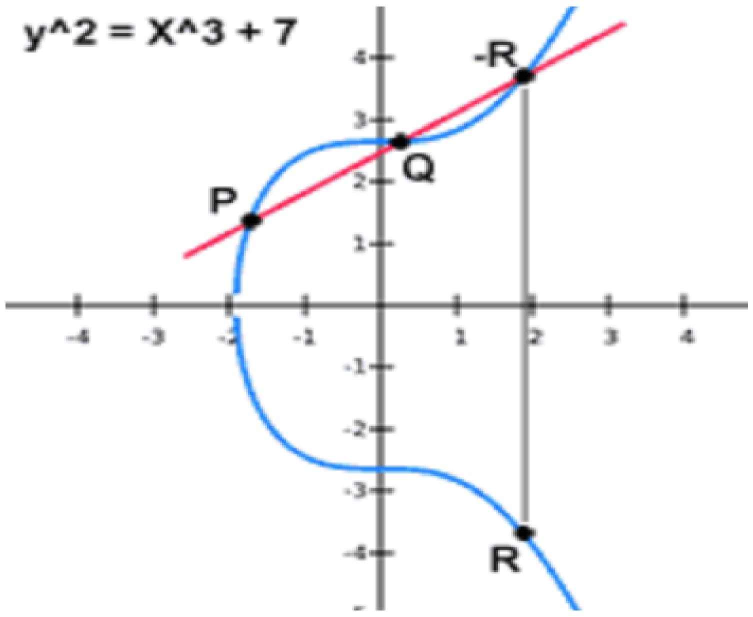

Figure 1. Koblitz curve secp256k1

- VeloxChain support for all EVM smart contracts (solidity and viper) and development tools, IDE, wallets and client software working with Ethereum chain.

To the best of our knowledge, it is very challenging or even impossible to have a one-size-fit-all protocol. And mentioned in the business white paper, we strongly believe that blockchain is the ideally technology to address many painful problems of sharing-mobility industry. VeloxChain technical philosophy implementation is the best balance between practicality and decentralisation. Therefore, we run Velox protocol to decentralize and connect sharing-mobility

applications. VeloxChain release PoS consensus, supporting EVM smart contracts (solidity and viper) and data privacy-preserving to solve needs for its user now. We will actively embrace in theoretically superior alternatives lacking a usable, production-ready implementation.

3. VeloxChain Infrastructure Technologies

3.1. Privacy-Preserving Solution with Zero-Knowledge Proofs

"Zero-Knowledge Proof are really mindboggling"

Sergey Brin, Google co-founder[1]

Blockchain is supposed to be a transparency machine in which anyone can join the network and as a result view all information on that network. If the confidential transaction is only between the two parties involved in the transaction, why do we need a consensus (blockchain)? In fact, **ZKPs and blockchains complement each other. A blockchain is used to make sure all parties can agree on some state which may or may not be encrypted, a trustless source to refer. ZKPs allow you to be confident about some properties of that state.** We take an example how VeloxChain DApps applying ZKPs. QiQ[2] is about building the world largest green transport system for everyone to create a trustless ecosystem and community ownership - the first use case in the VeloxChain business paper. Leveraging decentralised autonomous organization (DAO[3]) architecture, QiQ allow entrepreneur doing crowdfunding from crypto investors and then launch local electric fleet around the world. DAO is used as a blockchain crowd-funding tool and blockchain is used a payment and immutable accounting system to automatically distribute revenue sharing among stakeholders for examples if the softcap is reached, the local fleet operator will launch the s 30% to maintenance & fleet renewal, 20% QiQ platform, 20% DAO investors and 30% local fleet operator. It is a great blockchain application. **However, ZKPs privacy preserving smart contracts can make amount**

[1] https://www.trustnodes.com/2018/07/08/googles-sergey-brin-mining-ethereum-calls-zero-knowledge-proofs-mindboggling
[2] https://qiq.global/
[3] https://en.wikipedia.org/wiki/Decentralized_autonomous_organization

of crowd-funding privately until the campaign is over and if the crowd-funding are failed, only the campaign owner know the amount. By this way, ZKPs privacy preserving smart contracts may make QiQ entrepreneurs more confident and successful on the crowdfunding campaign.

Homomorphic encryption[4] is a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. The purpose of homomorphic encryption is to allow computation on encrypted data. In fact, Zero Knowledge Proofs (ZKPs) are to use different types of homomorphic encryption and are not new. They were first introduced in a paper "The Knowledge Complexity of Interactive Proof-System"[5] in 1985 by S Goldwasser, Silvio Micali and Charles Rackoff. ZKP is a mathematical expression which allows twos parties (a prover and a verifier) to prove that a state is true, without revealing any information about that thing apart from it being true. The design of a zero-knowledge proof must satisfy three properties:

- **Completeness**: if the prover and verifier are honest, the zero-knowledge proof always returns "true"
- **Soundness**: It is impossible for dishonest prover to convince a verifier to return "true"
- **Zero-knowledge**: the verifier learns nothing about the input (zero-knowledge)

In the cryptocurrency, there are few prominent blockchain projects using ZKPs such as ZCash, ZeroCoin.

3.2. Privacy-preserving collaboration in VeloxBiz

We believe that in the **VeloxBiz** platform, vehicle suppliers, fleet operators and retail service providers do not want disclosure of their commercial information. Merchants want privacy for things like supplier relationships and costs. Similarly, in the P2P context, riders and vehicle owners do not want to their data exposed without their permission.

[4] https://en.wikipedia.org/wiki/Homomorphic_encryption
[5] https://groups.csail.mit.edu/cis/pubs/shafi/1985-stoc.pdf

Therefore, confidentiality for data is a fundamental requirement for the VeloxChain ecosystem. We design anonymous collaboration layers to hide the real identity and a cryptographic search engine modelled by smart contracts to manage and share sensitive data.

3.2.1. Anonymous Collaboration Layer with Zero Knowledge Proof

Anonymous Collaboration Layer allows merchant A and merchant B to prove their collaboration anonymously by the merchant A obtains a credential from the merchant B so that at some later point in time, the merchant A is able to construct a non-interactive proof of his credential to perform authenticated transactions. The VeloxChain masternodes accept the request only if the attached proof is valid. The design of the component is inspired by the idea of Zerocoin[15] with Zero Knowledge Proof technique.

The following example describes how the anonymous collaboration layer works in practice. Let's suppose that there is a public bulletin board, one which is physical and everyone can access. There are two actors in the merchant market platform **VeloxBiz**: the bike-sharing fleet operator A collaborate with the scooter fleet operator B that the scooter-sharing fleet operator B provides his scooter asset to the bike-sharing fleet operator A and then the bike-sharing fleet operator A can provide multi-modal services (bike-sharing and scooter-sharing). In this scenario, the bike-sharing fleet operator A which plays the role of a data consumer and the scooter-sharing fleet operator B who plays the role of a data owner.

To produce a new credential for the bike-sharing fleet operator A, the scooter-sharing fleet operator firstly generates a pseudonym S for the party A and commits Susing a secure digital commitment scheme. The resulting commitment C can be opened using a random number r known by the party A. The party B pins C to the bulletin board, there is a set $S_C = (C_1, C_2, \dots C_n)$ of commitments in the board. At a later point, the party A is able to prove possession of such credential by producing two statements in zero-knowledge:

- He knows a commitment $C \in S_C = (C_{1,} C_2, \dots, C_n)$.
- He knows the opening r for the commitment.

The VeloxChain is used as a public bulletin board. Both the data owner and data consumer are able to access the public parts of the data stored on the blockchain. The public parts contain the commitments that we have described.

We now present a concrete construction using cryptographic accumulator proposed by Josh Benaloh[16], and later improved by Jan Camenisch[17]. The accumulator scheme comprises four algorithms:

- $AccumSetup(\lambda) \rightarrow params$. Generates two primes p, q, computes $N = pq$, sample $u \in QR_N$. Output (N, u) as the parameters.
- $Accumulate(C) \rightarrow A$.On a set of primes $C = \{c_{1,}\ldots, c_n\}$, outputs accumulator $A = u^{c_1 \ldots c_n} \ mod \ N$.
- $GenWitness(v, C) \rightarrow \omega$.Input a prime number $v \in C$, outputs a witness$K\omega = Accumulate(C - v)$.
- $AccVerify(\omega, v, A) \rightarrow \{0,1\}$.Verifies $A = \omega^v \ mod \ N$The security of the scheme is based on the harness of Strong RSA and Discrete Logarithm assumptions.

The description of the anonymous authority layer consists of four algorithms:

1. $Setup(1^\lambda) \rightarrow params$. On the input parameter λ, run the algorithm $AccumSetup(1^\lambda)$to obtain (N, u). Generate primes p, q such that $p = 2^w q + 1$for $\omega \geq 1$.Let G be the subgroup of $Z_q{}^*$and select two random generator g, h such that $G = [g] = [h]$.
2. $GenCred(S, params) \rightarrow (c, skc)$. Given pseudonym $S \in Z_q{}^*$, select a random $r \in Z_q$ and compute $c \leftarrow g^S h^r$such that c prime and $c \in [A, B]$, where $2 < A$ and $B < A^2$. Set $skc = r$ and output (c, skc), submit $c$to the blockchain.
3. $ShowCred(params, S, c, skc, Sc) \rightarrow \pi_S$ Given data consumer pseudonym S, a credential c and its secret key skc, compute $A = Accumulate\,(params, S_c)$ and $\omega = GenWitness\,(params, c, S_c)$ and output the following proof of knowledge:

$$\Pi_s = ZKSoK\{\,(c, w, r, S) : AccVerify((N, u), A, c, \omega) = 1 \wedge c = g^S h^r\}$$

4. $VerifyCred(params, \pi, S_c)$. Given a proof Π_S, and the public set of credential S_C, first compute $A \rightarrow Accumulate\,(params, S_C)$, then verify that Π_S is the aforementioned proof of knowledge on c, S_c. if the proof verifies successfully, output 1, otherwise output 0.

The zero-knowledge proof which appears in step 3 of the scheme is a non-interactive proof that only requires one round of communication. Camenisch presents an interactive zero-knowledge proof of knowledge in which an accumulator contains a committed value. The construction of the non-interactive proof in step 3 leverages a Fiat-Shamir transform on the interactive proof. This is shown in the process flow diagram on the following page.

The $Setup$ algorithm is performed by the data owner to generate system parameters. Next, when the data consumer wishes to obtain a credential for data access, he sends a request to the data owner together with his pseudonym S. At this point, the data owner runs the $GenCred$ routine on this input S to generate a digital commitment and its secret key skc.

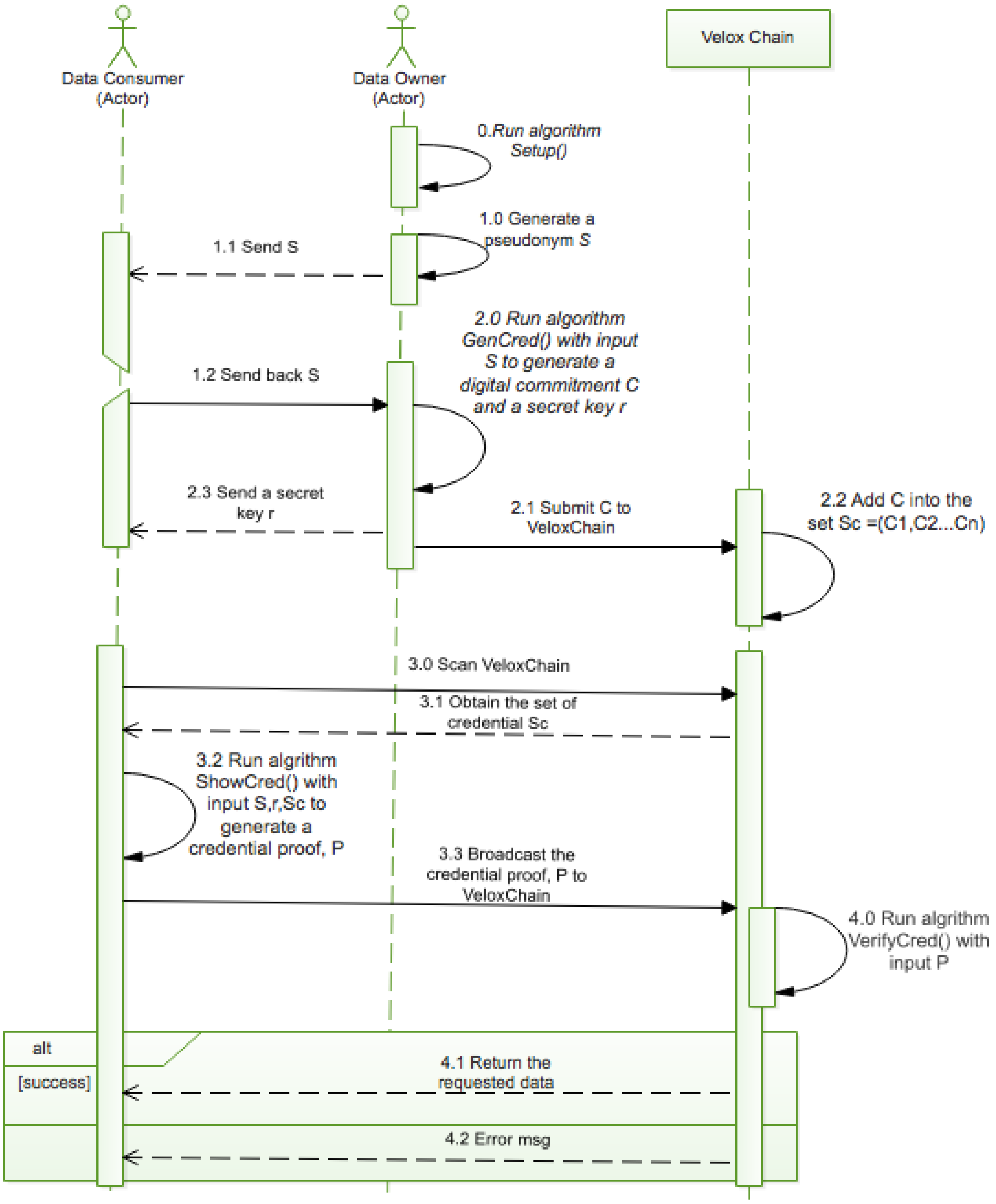
Data Consumer (Actor)
Data Owner (Actor)
Velox Chain
0.Run algorithm Setup()
1.0 Generate a pseudonym S
1.1 Send S
2.0 Run algorithm GenCred() with input S to generate a digital commitment C and a secret key r
1.2 Send back S
2.3 Send a secret key r
2.1 Submit C to VeloxChain
2.2 Add C into the set Sc =(C1,C2...Cn)
3.0 Scan VeloxChain
3.1 Obtain the set of credential Sc
3.2 Run algrithm ShowCred() with input S,r,Sc to generate a credential proof, P
3.3 Broadcast the credential proof, P to VeloxChain
4.0 Run algrithm VerifyCred() with input P
alt
[success]
4.1 Return the requested data
4.2 Error msg

Figure 2. Sharing Data in the Anonymous Collaboration Layer

When the data consumer wishes to show his credential, he first scans through the VeloxChain to obtain the set S_c consisting of all credential issued by the data owner. He then runs the $ShowCred$ routine to generate a credential proof, and broadcasts it to all masternodes for verification. The masternodes also collect the set of credentials in the blockchain and validate the proof using the $VerifyCred$ algorithm. The credential certification is accepted if the last routine outputs 1.

The data consumer and the blockchain nodes are both required to compute $A = Accumulate\ (params, S_c)$ which requires a linear scan of the blockchain data. The complexity of the protocol increases linearly with the number of registered data consumers.

With this robust anonymous collaboration layer, the bike-sharing fleet operator A and the scooter-sharing fleet operator B can work together and exchange data with each other without exposing her/his identity address. It means that it is very difficult to trace the owner of collaboration transactions in VeloxChain but it is also easy to prove someone to participate in the collaboration.

3.2.2. Private Keyword Search Engine

In the Velox ecosystem, we provide the distributed data storage layer. IPFS[18] is used to store public and plain-text data while sensitive data are encrypted and stored in the distributed storage.

In order to support sharing-mobility DApps to easily develop and manage encrypted sensitive data, VeloxChain provides a built-in encrypted data search modelled by smart contracts. The crucial requirement of such a search must be fast return of results without exposing private data. Our solution is that the meta-data (i.e. the fingerprint) of the encrypted data are stored in the VeloxChain and only authorized clients are permitted to use it to conduct searches. The authorization process is done by the Anonymous Collaboration Layer presented in the section 3.2.1 An access key is computed using a hash function (i.e. The fingerprint of the data). The VeloxChain does not store the actual data content, however, it maintains the access key data so that data consumers are able to link real data to the distributed storage layer using blockchain.

We denote EKS as the encryption scheme that supports keyword search. The data owner appends a list of EKS ciphertext of each keyword to the access key and stores it in the blockchain layer. A data D with keywords $W_1, W_2, \ldots, W_n$ is stored in the blockchain layer under the structure: $H(D)\ ||EKS(W_1)\ ||\ldots||EKS(W_n)$. An authorized data consumer is able to produce a certain trapdoor Γ_ω that enables a smart contract to test on each data entry whether one of the keywords associated with the access key (eg- the document) is equal to the word W. Given a trapdoor and EKS ciphertext, the blockchain nodes can only test whether $W = W'$, and nothing else.

A typical keyword search cryptosystem consists of four general algorithms:

1. $KeyGen$:generates cryptosystem key.
2. $Trapdoor$: produces trapdoor T_W for a keyword W
3. $Encrypt$:produces a EKSciphertext for keyword W
4. $Test$: tests whether keyword in the trapdoor is matched to the EKS ciphertext.

In the VeloxChain ecosystem, an additional algorithm is required for the data owner to produce a secret search key for the data consumer. We denote that algorithm $KeyDerive$. Three algorithms $KeyGen, Encrypt$and $KeyDerive$ are performed by the data owner, while $Trapdoor$ is run by the data consumer to generate a trapdoor, and finally, the $Test$ algorithms is performed by smart contracts or the blockchain peers. We modify the protocol proposed by Raluca Ada Popa[19] to adapt to our environment.

We start the protocol description by reviewing a few concepts related to bilinear maps. We will use the following notation: $G_1$and $G_2$are two (multiplicative) cyclic groups of prime order $p, g_1$is a generator of G_1 and $g_2$is a generator of G_2. A bilinear map is a map $e: G_1 \times G_2 \rightarrow G_T$with the two following properties: $(1)\, Bilinear: \forall u \in G_1, v \in G_2$ and $a, b \in Z$,then $e(u^a, v^b) = e(u,v)^{ab}$, and $(2)\, Non-degenerate: e(g_1, g_2) \neq 1$.

We denote $H: \{0,1\}^* \rightarrow G_1$ and $H_2: G_T \times G_T \rightarrow \{0,1\}^*$ to be two random oracles, and g_1, g_2, g_T are respectively the generators of groups G_1, G_2, G_T. The private keyword search system consists of five algorithms as the follows:

1. $KeyGen: k \leftarrow Z_p$.
2. $KeyDerive(k, s):\ k_s \leftarrow g^{\frac{k}{s}}$.
3. $Trapdoor(w, s): T_w \leftarrow e(\, H(w)^s, k_s)$.
4. $Encrypted(k, w): Random\ r \leftarrow G_T. Output: c = (r, H_2(r, e(H(w), g_w)^k)$.

5. $Test: Parse\ c = (r, h)$. Test whether $H_2(r, tk) = h$.

The data owner generates a secret key k for keyword encryption EKS, and derives keys for the data consumers. Each data consumer poses a secret $s \in Z_p$, which can be generated by a mapping from his pseudonym with the data owner. Using $k$and s, the data owner computes a search key k_s for the data consumer so that later he can use it for trapdoor construction.

The correctness of the protocol follows the two equations:

$tk = e(H(w)^s, g_2{}^{\frac{k}{s}}) = e(H(w), g_2)^k$, and $H_2(r, tk) = H_2(r, e(H(w), g_2)^k)$

The above scheme has data hiding and token hiding properties. Data hiding (privacy) requires that the semi-honest adversary is not able to distinguish between ciphertexts of two values not matched by some token. Token hiding (privacy) requires that the adversary cannot learn the keyword that one searches for. <u>The complexity of the protocol increases linearly with the number of data sets stored in the VeloxChain ecosystem</u>.

3.3. VeloxChain Consensus - Proof of Stake

It is widely accepted that a blockchain-based system is as secure and robust as its consensus model. The best consensus model will be one which best balances the following three following factors:

- Decentralization: Any node freely participates in processing transactions, and publishing a block without the use of a central authority or service.
- Security: The system has to prevent double- spends, keep data in sync. There

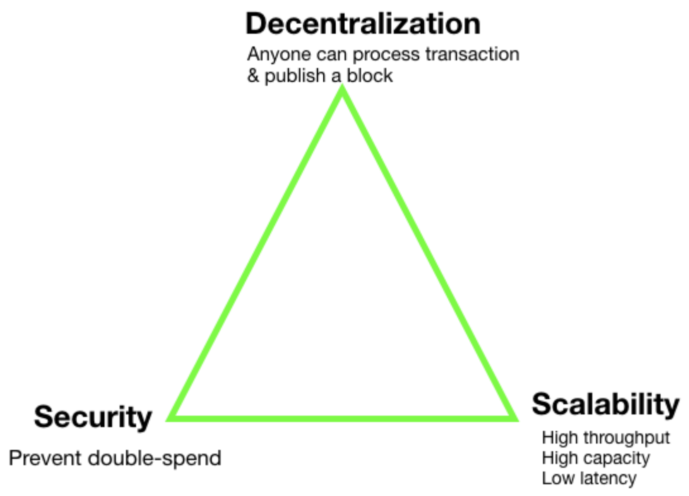

Figure 3. Three crucial factors of blockchain consensus algorithms

are no conflicts when data get merged. All nodes see same data at the same time.

- Scalability: The system has to provide sufficient transaction throughput to serve enterprise-scale or even planet-scale needs, especially when the network size increases.

Following is a table listing different types of consensus algorithms, and highlighting the strengths of each.

	Types of Consensus				
Features	**PoW**	**PoS**[9]	**PoET**	**BFT * and variants**	**Federated BFT**
Blockchain Type	Permission-less	Both	Both	Permissioned	Permission-less
Blockchain	BitCoin, Ethereum	TomoChain, EOS, Omisego	Intel Ledger	Hyperledger Fabric	Ripple & Stellar
Transaction finality	Probabilistic	Probabilistic	Probabilistic	Immediate	Immediate
Transaction rate	Low	High	Medium	High	High
Scalability of peer network	High	High	High	Low	High
Trust model	Untrusted	Untrusted	Untrusted	Semi-trusted	Semi-trusted
Adversary Tolerance	<= 25%	Depends on specific algorithm used	Unknown	<= 33%	<=33%

* Note: BFT is Byzantine Fault Tolerance

Figure 4. A comparison of blockchain consensus mechanisms

VeloxChain uses the PoS with on-chain governance philosophy of design which achieves the best balance of three factors above. With PoS, VeloxChain has a 2-second block time with over 1000 tx/s and provides a good security by leveraging the economic-game theory.

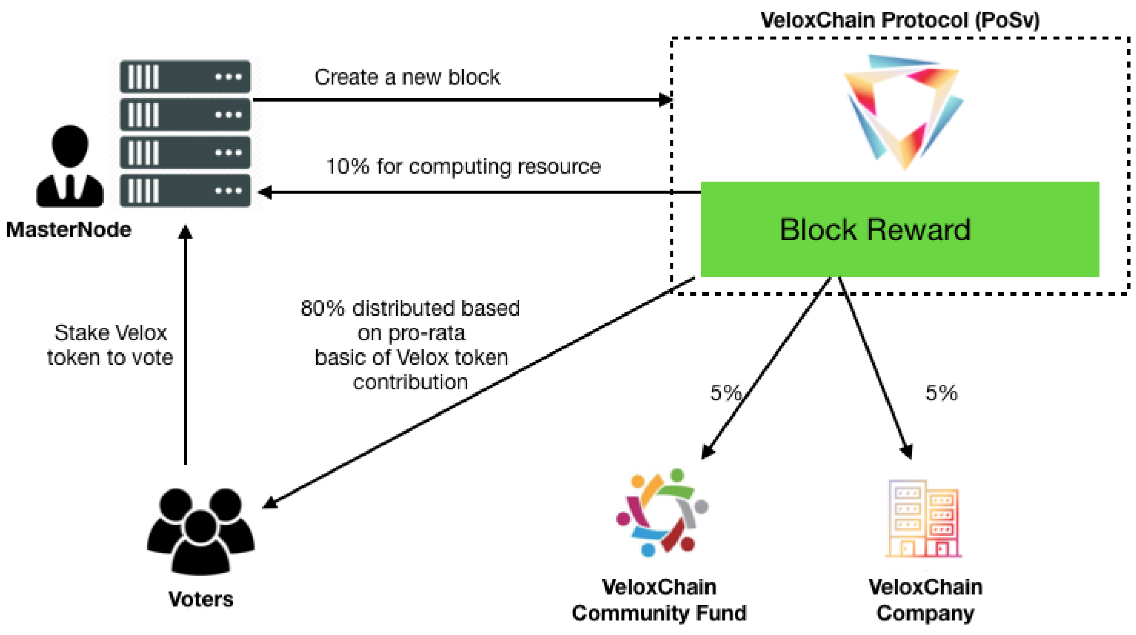
VeloxChain Protocol (PoSv)
Create a new block
10% for computing resource
MasterNode
Block Reward
Stake Velox token to vote
80% distributed based on pro-rata basic of Velox token contribution
5%
5%
Voters
VeloxChain Community Fund
VeloxChain Company

Figure 5. Token distribution among stakeholders in Veloxchain PoS

We believe that PoS with on-chain governance features provide more decentralized ecosystem and fairness distribution of value among stakeholders who takes responsibilities for development of the ecosystem. For example:

- **Masternodes** deposit Velox token to become a full-node to create, verify and validate a new block. Masternodes earn a significant portion of block reward and transaction fees.
- **Voters** are Velox token holders who participate in governing the chain by deposit token and voting someone becoming a masternode. As the result, the voters will also receive Velox tokens when their candidates become the masternodes and proceed transactions

Moreover, VeloxChain aims to connect sharing-mobility services and thus a small portion of block reward are distributed to the VeloxChain community wallet which will support sharing-mobility applications and incentivise users to use sharing-mobility service.

4. VeloxChain DApp’s Architecture

Overall, there are five main layers in Velox Ecosystem:

- **Application Layer**: Besides VeloxGo and Velox Biz, there are significant numbers of sharing-mobility applications built on top Velox ecosystem such as bike-sharing, scooter-sharing, car-sharing. VeloxChain will provide SDK and firmware libraries to allow existing sharing-mobility application integrating easily.
- **Middle Layer**: there are SDKs and APIs allow third-party systems, IoT and smart vehicles to interact easily with the Velox Ecosystem.
- **Service Layer**: are distributed system hosted by VeloxChain or mobility providers. The service layers play as a “bridge” communication among applications and decentralised layer. While on-chain communication allows merchants to communicate by filtering event log or interacting directly with smart contracts; the service layer enable off-chain peer-to-peer communications between merchants such as the scooter-sharing app can broadcast a message to other merchants to look for alternative transport mode (bike, car or even their competitor scooter) if that particular company has no scooter available, it is a win-win deal and VeloxChain smart contracts ensure revenue-sharing among merchants automatically.In the bridge service layer, there are some core components:
    - Cache and indexing data module: allow the bridge server to sync data from decentralized layer and then index the data to provide fast search engine.
    - Order Queuing module: allow stakeholders to communicate with each other such as booking, accepting or cancelling a request. The queuing module can handle communication between users and vehicle providers, or among sharing-mobility service merchants

    - Meta Transaction Relay module: allow service layers to fuel gas of transactions or forward transactions to the decentralised layer.
    - Pricing and payment module: allow sharing-mobility service to use fiats or cryptocurrencies as payment methods.
- **Abstraction Layer**: are zero-knowledge of proof, keyword privacy-preserving search, payment channel smart contract- frameworks which enables seamless integration into the Velox Chain.
- **Decentralisation Layer**:
    - Public blockchain protocol for the sharing-mobility industry: VeloxChain support with Ethereum Virtual Machine (EVM) smart contracts (solidity and viper), use Proof-of-Stake consensus with over 1000 tx/s and 2 second block-time and have data privacy-preserving features, key management feature.
    - Distributed storage layer: IFPS hosts to store public plain-text data while sensitive data are encrypted and stored in the distributed storage system.

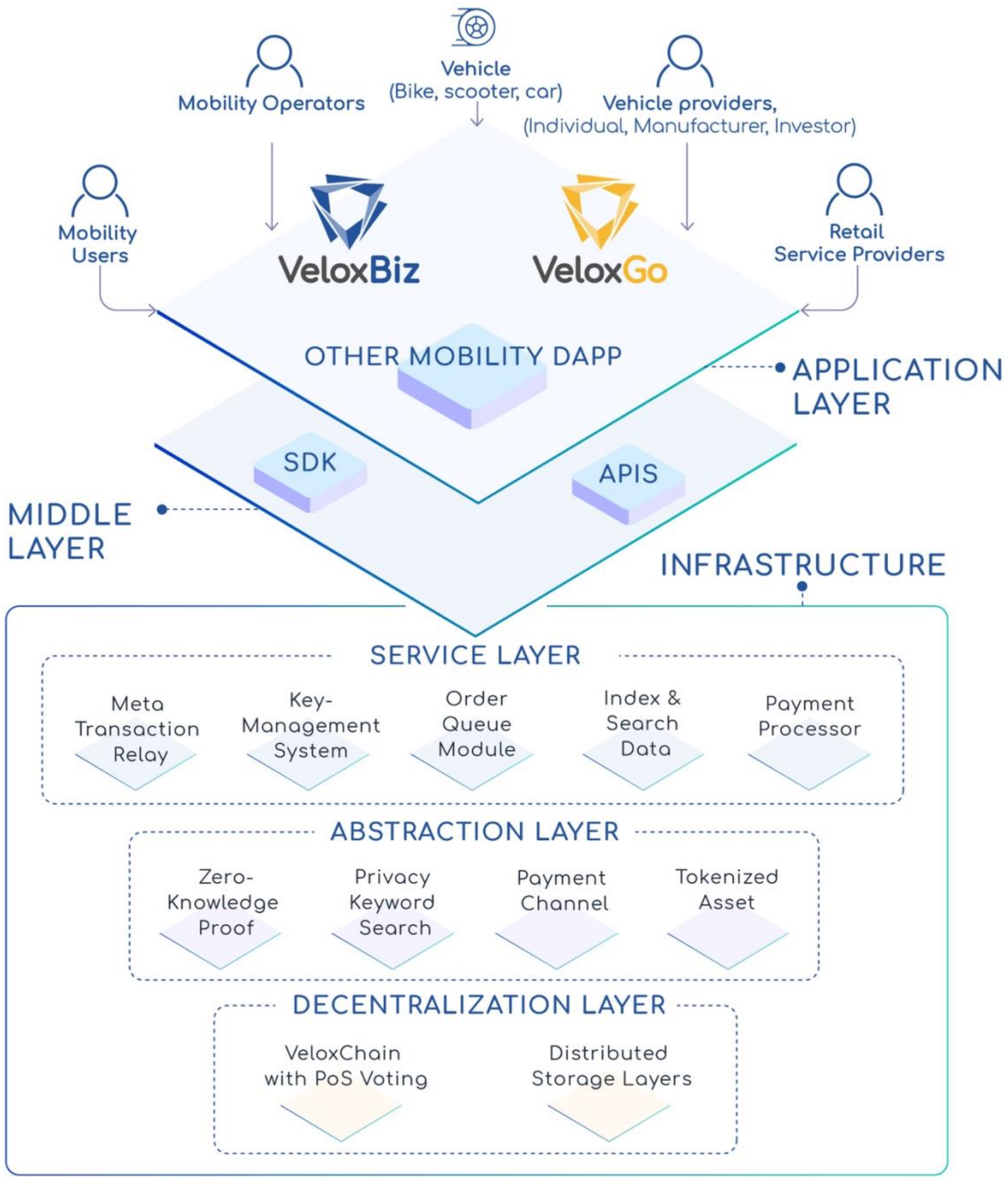

Figure 6. High-level Architecture of VeloxChain Ecosystem

4.1.1. Meta Transaction Relay Protocol

All on-chain transactions on blockchain are paid using transaction fees known as *gas*. Many people, however, have little-to-no knowledge about blockchain and hold no Velox token. Therefore, without meta transaction relay service, we can imagine the customers like Join use a scooter-sharing DApp from merchant B:

1. Join know about the scooter-sharing DApp through his friend and download it to try.
2. Join try to create an account and book a scooter from merchant B; but he realizes that he has no Velox token to perform on-chain transactions.
3. Join has to go to his favourite exchange, doing KYC and wait for few days to buy some Velox token. And then he is able to perform on-chain transactions.

Consequently, it is a terrible user experience to force all users to buy and then hold token to use sharing-mobility services on VeloxChain. Meta Transaction Relay aims to solve this problem by allowing a third-party to fuel gas of transaction (fund) and then replay the transaction for the users.

How the meta transaction relay magic works in the practical example. Firstly, Join download the scooter-sharing DApp from merchant B. Although Join holds no token, he can sign the transaction and send this transaction to a relayer server which probably to hosted by the merchant B. This relayer server can pay the gas for this transaction and then forward it to Velox Chain through the *relayMetaTx*[22] protocol. As a result, Join does not require to hold token and the merchant B is able to pay his users on-chain transactions.

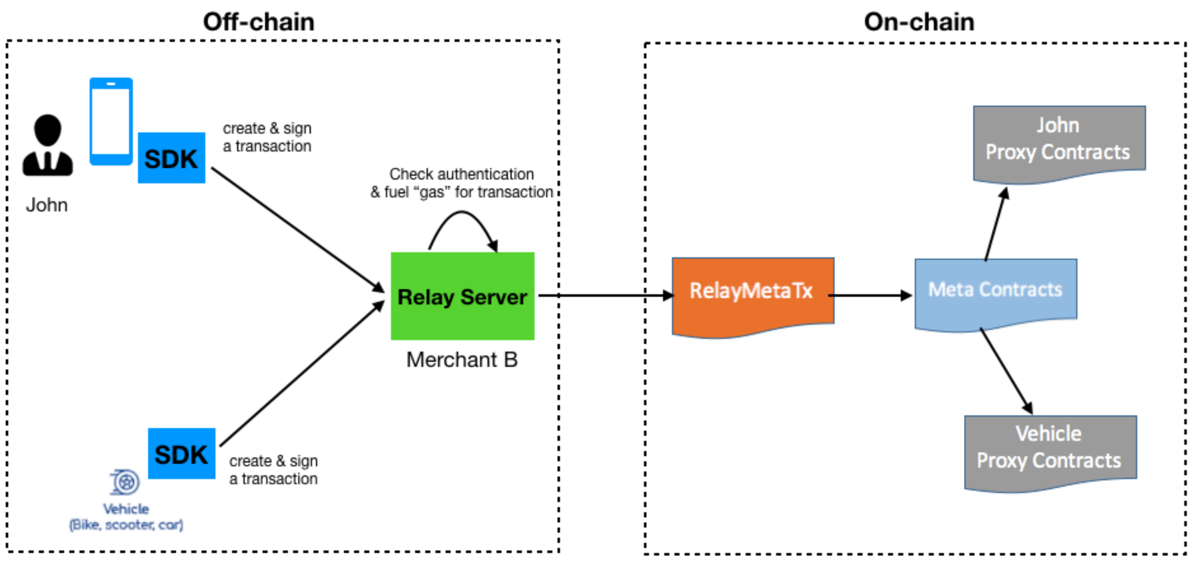
Off-chain
On-chain
John
SDK
create & sign a transaction
Check authentication & fuel "gas" for transaction
Relay Server
Merchant B
SDK
create & sign a transaction
Vehicle (Bike, scooter, car)
RelayMetaTx
Meta Contracts
John Proxy Contracts
Vehicle Proxy Contracts

Figure 7. Meta Transaction Relay Service.

4.1.2. Payment Channel

The payment-channel[20] network allows us to transfer token near-instantly and with low-fees by using digitally signed and hash-locked[21] transfers. In the payment channel, users have to setup an on-chain deposit to open a payment channel smart contract, and then users can perform token transfers instantaneously, without limit, as long as the net sum of their transfers does not exceed the deposited tokens. Finally, users have to close the payment channel by calling functions $cooperativeClose\ (\quad) or\ uncooperativeClose\ ()$ in the smart contract $TransferChannels.sol$ [22] which requires on-chain transactions. VeloxChain leverages multi-signature smart contracts and payment-channel technology to enable automatically revenue-sharing models among stakeholders.

4.1.3. Vehicle Ownership and Tokenization from ERC721

VeloxChain has developed a Vehicle-Ownership protocol to share ownership information. Vehicle owners can prove their ownership and report the location of stolen the smart vehicle by referencing immutable records on the blockchain. When a smart vehicle is stolen, the police are able to see the latest position to respond instantly. Potentially, insurance companies could automate claim processing for such losses through a smart contract running on the blockchain.

We have often heard how the Ethereum ERC721 smart contract is non-fungible, meaning that tokens of the same class or contract can hold a different value. The Cryptokittie[24] project is one interesting application. Cryptokitties are unique (and collectible) assets because each is tokenized with a different ERC721, having a different value.

For this reason, we developed the Vehicle-Ownership protocol[25] using ERC721 tokens from OpenZeppelin[26]. Each ERC721 represents a different vehicle, and the value of the Vehicle-Ownership token is based on the metadata of token including vehicle technical information, picture, owner's information, supplier, and manufacturer. Ownership is determined by an array of token *indexes* or *ids* that is mapped to owner address. The total supply of Vehicle-Ownership tokens is the length of array $allTokens$. The number of vehicles registered in the VeloxChain ecosystem is virtually unlimited, as the maximum number of any given type of ERC721 tokens, based on the unsigned integer storage type, is $2^{256} - 1$.

When a vehicle owner registers a new vehicle, the system will generate a random integer as a new token id and then call $addTokenTo$from the VehicleOwnershipProtocol.sol to issue a new Vehicle-Ownership token.

Each Vehicle-Ownership token holds metadata which is the URI of the vehicle information record, such as manufacturer, vehicle technical information, images storing in the distributed storage layer.
Whenever users create a new Vehicle-Ownership token, the matching engine scans through all tokens using our searching engine (described in section 2.2.1); and if the matching engine finds any Vehicle-Ownership token with the same metadata, the system suggests users merge or transfer their token between users.

Consider this simple example: Alice bought a Volata smart bicycle from Bob. This bike has a computer chip IMEI: VOLATA123456789. With the bike, Alice obtains a new Vehicle-Ownership token. But the matching engine detects that Alice's token has the same metadata (i.e.- IMEI code) as another vehicle-ownership token held by the bike manufacturer. The system notifies Alice and the manufacturer to merge these Vehicle-Ownership tokens. This could be accomplished by Alice burning the token received from Bob and the manufacturer sending Alice the token in its possession, by a multi-signature contract.

5. Reference

[2] Ethereum https://en.wikipedia.org/wiki/Ethereum

[3] BMW, GM, Ford and Renault launch blockchain research https://techcrunch.com/2018/05/02/the-mobility-open-blockchain-initiative-bmw-gm-ford-renault/

[4] "Ethereum Next 12 Months" by Vitalik Buterin
https://www.youtube.com/watch?v=jJt3yag96fU

[5] Ethereum's Casper and Sharding Delay
https://www.trustnodes.com/2018/07/08/ethereums-casper-sharding-delay-disappoints-can-2020-deadline-reached

[6] EOS smashes blockchain records with transaction speeds faster than visa.
https://blokt.com/news/eos-smashes-blockchain-records-with-transaction-speeds-faster-than-visa

[7] Elliptic Curve Digital Signature Algorithm.
https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

[8] SEC 2: Recommended Elliptic Curve Domain Parameters
http://www.secg.org/sec2-v2.pdf

[9] Proof-of-stake https://en.wikipedia.org/wiki/Proof-of-stake

[11] Sharding in Zilliqa blockchain https://docs.zilliqa.com/whitepaper.pdf

[12] Loom Network - highly scalable DPoS sidechains to Ethereum
https://medium.com/loom-network/dappchains-scaling-ethereum-dapps-through-sidechains-f99e51fff447

[13] Cosmos - a decentralized network of independent parallel blockchains
https://cosmos.network/docs/resources/whitepaper.html#introduction

[14] Polkadot: Vision for a heterogeneous multi-chain framework
https://polkadot.network/PolkaDotPaper.pdf

[15] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin
http://zerocoin.org/media/pdf/ZerocoinOakland.pdf

[16] Josh Benaloh, Michael de Mare https://www.microsoft.com/en-us/research/wp-content/uploads/1993/01/owa.pdf

[17] Jan Camenisch, Anna Lysyanskaya
http://cs.brown.edu/~anna/papers/camlys02.pdf

[18] IPFS - InternPlanetary File System
https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

[19] Raluca Ada Popa and Nickolai Zeldovich
https://people.csail.mit.edu/nickolai/papers/popa-multikey-eprint.pdf

[20] Payment Channel Technology in Raiden- https://raiden.network/101.html

[21] Hashlock - https://en.bitcoin.it/wiki/Hashlock

[22] Smart contract *RaidenMicroTransferChannels.sol*
https://github.com/raiden-network/microraiden/blob/master/contracts/contracts/RaidenMicroTransferChannels.sol

[23] Velox Meta Transaction Relay
https://github.com/VeloxChain/Velox_Meta_Transaction_Relay_Protocol

[24] Cryptokitties https://www.cryptokitties.co/

[25] Vehicle-Ownership protocol
https://github.com/VeloxChain/VehicleOwnershipProtocol