www.platon.network

PlatON

Privacy-Preserving Computation in PlatON Version 1.0

LatticeX Foundation /

—

July 15, 2021

Abstract

Privacy-preserving computation is the key component and ability of PlatON and Alaya networks, which are open-source projects under LatticeX Foundation. LatticeX Foundation aims to solve three main problems in the blockchain community, including privacy, storage and key management. This article describes the underlying cryptographic components and potential capabilities to support privacy-preserving applications in these two networks.

Contents

1 / Overall Framework

Blockchain community has grown very rapidly these years, and plenty of applications arere-built in decentralized systems. However, many urgent problems still exist and shall be solved in blockchain systems. This article aims to consider privacy, storage and key management, which are widely discussed in the community.

Dating back to the design of Bitcoin, all participants maintain the network by storing copies of all the data. Although it provides a potential method to achieve “consensus”, the transaction data is public to all the participants. Ethereum introduces smart contract that extends payment transactions into any computable functions. Developers could design their business logic using smart contract, once the condition is satisfied, the smart contract will be run automatically. Automatic execution essentially means that each node of the network executes the smart contract repeatedly in local. Therefore, the input, output and description of the function should be public to all participants.

Developers now realize privacy is becoming one of the most important issues when building financial business among enterprises with blockchains. The privacy of the transfer amount, the anonymity of payer and payee, and the privacy of the business logic (i.e., the function described in smart contract) should be protected in most scenarios. LatticeX Foundation aims to provide a privacy-preserving infrastructure for decentralized business among financial institutes.

As the name of blockchain, each block containing multiple transactions is chained to another with a cryptographic hash function. The one-wayness of the hash function and append-only structure of blockchain make it computationally hard to tamper data stored. However, as transactions continue to grow, so does the amount of data that needs to be stored on the blockchain. At the time of writing this manuscript, Bitcoin and Ethereum store hundreds of gigabyte amount of data, and the size is still increasing. It takes days on a typical laptop as a full node to synchronize and verify these historical transactions. The underlying consensus protocol of PlatON/Alaya is based on proof of stake (PoS). In a PoS system, the validators often run full nodes in a cloud provided by a third party, and almost over 50% cost of cloud services comes from storage. With the increasing of blockchain data, the cost of validator arises. Therefore, reducing the size of storage is important for validators to reduce cost and further maintain the security of the network.

LatticeX Foundation proposes a framework, as shown in Figure 1, to deal with privacy and storage issues in PlatON/Alaya. This solution or plan is based on the architecture of PlatON 2.0 (see the white paper [28]). The vision of PlatON 2.0 is to build a decentralized and collaborative AI network and global brain to drive the democratization of artificial general intelligence.

A three-layer network is considered in PlatON 2.0. The first layer is Consensus Layer. Consensus network is a decentralized network composed of blockchain nodes, which are connected to each other through P2P protocols and can be consensual through consensus protocols in an environment where no one needs to be trusted. On the blockchain network, smart contracts can be executed automatically. The second layer is Privacy-Preserving Computation Network. Data nodes and computing nodes could connect to this layer and publish data and contribute computing power. Through smart contracts on the blockchain, a decentralized sharing and trading marketplace for data, algorithms and computing power can be built. Based on the cryptographic economics it will form an effective incentive mechanism to motivate more data, algorithms and computing power to join the network. The third layer is Collaborative AI Network, in which multi-agent systems and AI agents can operate independently, and finally form autonomous AI networks.

One could take the solution of this article as advanced properties of PlatON 2.0 in layer 1 and layer 2. It is designed dedicatedly to solve privacy and storage problems mentioned above.

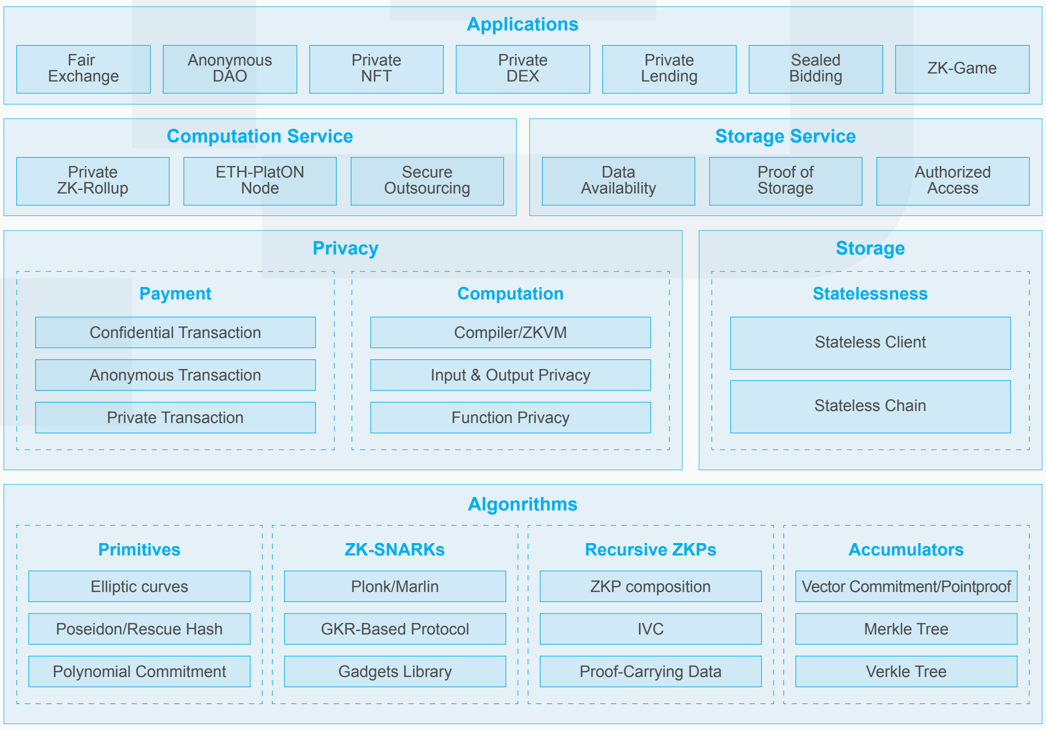

Figure 1: The Overall Framework of Privacy-Preserving Computation.

Four levels are considered in this framework. The first level contains necessary cryptographic algorithms, especially related to zero-knowledge proofs. The second level is built on layer 1 in PlatON 2.0, it focuses on providing on-chain capabilities to solve the two problems above. The introduction of cryptographic tools will reduce the entire performance inevitably, for which the raw transactions need to be stored in some place anyway, instead of on the blockchain. The third level, which is built on layer 2, aims to provide computation and storage services off the chain, and thus we call it off-chain services. The fourth layer is built for privacy-preserving applications. Standard APIs, templates and protocols are created for different applications. More details of each level will be described in the coming sections.

In addition, one common requirement for all applications is to manage the private key securely. This article also provides a relatively independent solution for key management service in Section.[6]

2 / Algorithms

This level contains cryptographic algorithms to support upper level solutions. It have four classes including primitives, ZK-SNARKs, recursive zero-knowledge proofs and accumulators. Note that most of these algorithms are based on elliptic-curve cryptography, which turns out to be nonquantum safe. Although we believe that the power of quantum computers now poses no substantial threat to cryptography in a decade, we continue to maintain an ongoing interest in post-quantum cryptography. The research of quantum-resilient alternatives is very active in the academic community, we will migrate to the post-quantum era gradually when these algorithms are ready.

2.1. Primitives

Primitives are building blocks to construct advanced cryptographic algorithms and solutions.

Elliptic Curves. Elliptic-curve cryptography enables smaller public key size and fast operations. Inheriting from Ethereum the Secp256k1 curve is supported natively. Faster Curve25519 is also integrated to generate fast zero-knowledge proof, e.g., range proofs, based on discrete log problems. Another type of curves are pairing-friendly ones, including BN254 [1] and BLS12-381. These two curves are widely used in ZK-SNARKs algorithm and related pairing-based constructions such as BLS signature and verifiable function. Due to its versatility, pairing-friendly curves will be unified with BLS12-381 in the near future.

Hash Functions. Standard hash function used in blockchain is SHA256. Although the computation of SHA256 is extremely fast, it is not compatible with zero-knowledge proof algorithms.This is because SHA256 consists of well-designed Boolean circuits, but almost zero-knowledge proof algorithms are designed for arithmetic circuits. In theory, one could transfer SHA256 circuit into arithmetic in a field, but the performance of proof generation of the underlying ZKP algorithm is not satisfactory.

Zero-knowledge friendly hash functions play an important role to improve the performance of proof generation time. These functions are directly defined on arithmetic

[1] Someone may call it BN256

circuits, which naturally compatible with most ZKP algorithms. Poseidon [22] and Rescue [1] are two of the candidates. However, it is worth pointing out that these hash functions are newborn ones, which lack of sophisticated cryptographic analysis and test of time. We keep a constant eye on the progress of these algorithms, and cautious approach to the use of these algorithms is recommended.

Polynomial Commitment. A polynomial commitment scheme allows a committer to commit a polynomial with a short string that can be used by a verifier to confirm claimed evaluations of the committed polynomial. Polynomial commitment has a wide range of applications. It is a core building block of ZK-SNARKs, vector commitment and verkle tree et al. We recommend the efficient construction proposed in [24], which is also named as the KZG polynomial commitment. The KZG polynomial commitment has constant size (one single element), the overhead of opening one and multiple evaluations requires only a constant amount of communication, which is desirable in most applications.

However, the advantages of KZG polynomial commitment come at a price. It needs a trusted setup procedure, which is not acceptable in a decentralized network and applications. An ongoing activity named as Lumino [27] is a MPC ceremony to generate all the parameters in a distributed manner, and the resulting parameter is trustless as long as at least one participant is honest.

2.2. ZK-SNARKs

ZK-SNARKs are core components of the algorithm level. Essentially, ZK-SNARKs are noninteractive proof systems that could prove any NP problems in zero-knowledge. The proof size and verification time are very small, but the proving time is relatively large. These properties are very suitable for blockchain systems. Small proof size and verification time are beneficial to broadcast and verify the proof on-chain, and the proving procedure could be done off-chain.

Plonk [17] and Marlin [12] are preferable in our systems. Besides of the short proof and verification time, these two systems have another attractive property. Both of them only require a universal structured reference string to setup the system. This string plays a vital role to enjoy short proof size and verification time, and could be generated in a distributed way by running a very simple and efficient MPC protocol. Further, this string is independent of the applications (not like Groth16 [23]), and could be updated at any time by any participant. The very recent activity Lumino [27]is an MPC ceremony to generate this string for Plonk. We note that there exists ZK-SNARKs or other types of proof systems that do not need trusted setup. However, the concrete performance of most these algorithms seems not satisfactory for most complicated applications.

GKR-based algorithms [19] are another type of ZK-SNARKs. Although the concrete proof size and verification time are not as good as Plonk and Marlin, the proving time is attractive in other off-chain applications. Recent progresses show their huge potentials[35],[38],[37],[32]. This type of algorithm could even prove complicated statements such as deep learning models, which is the-state-of-the-art ZK-SNARK scheme in this direction.

In order to reduce the bar of developing with ZK-SNARKs, gadgets libraries under different algorithms are provided. In addition to facilitating development, these basic components allow for targeted optimization of them.

2.3. Recursive ZKPs

Recursive ZKPs are proofs of proofs. They essentially prove that another zero-knowledge proof is valid. These proof systems are very powerful in solving the scalability and storage issues in blockchain. However, the performance of all existing schemes is not satisfactory. We are very optimistic about the research in this area and have great confidence in the subsequent performance improvement.

ZKP Composition. Zero-knowledge proofs are usually used to protect the anonymity and confidentiality of transactions. However the introduction of ZKP will inherently reduce the performance of blockchain, this is because the verification procedure will take tens of milliseconds and is much larger than a single signature verification. Using ZKP composition, one could batch many (say 100) transactions protected by ZKP, and prove that “the zero-knowledge proof of these 100 transactions are valid”, then the resulting proof actually convince participants all the 100 transactions in a single verification. This again improves the amount of transaction per second (TPS) of the blockchain system.

Incrementally-Verifiable Computation. IVC proposed by Valiant [34] is a general case of ZKP composition. IVC splits a long sequential computation between different parties such that every party performs a small part of the computation and passes it on to the next party. Together with the current state of their computation, parties should include a proof that the computation was performed correctly, not only in the last step, but in its entirety.

A simple example is depicted in Figure 2 . The computation is split into F_0 , F_1 , F_2 in sequence and distributed to 3 parties. IVC enables anyone to verify the final result and the intermediate computations are correct, without compute the functions again. The basic idea of IVC is as follows. The first party computes F_0 on the input and provides a proof to convince the second party that the computation is correct. The second party computes F_1 on x_1 and provides a proof which says “I have a valid proof that $x_1 = F_0 (x_0)$”. The third party acts similarly, and provides a proof says “I have a valid proof that $x_2 = F_1 (x_1)$”. Therefore, one could verify the validity of π_2 to believe that all the computations are correct.

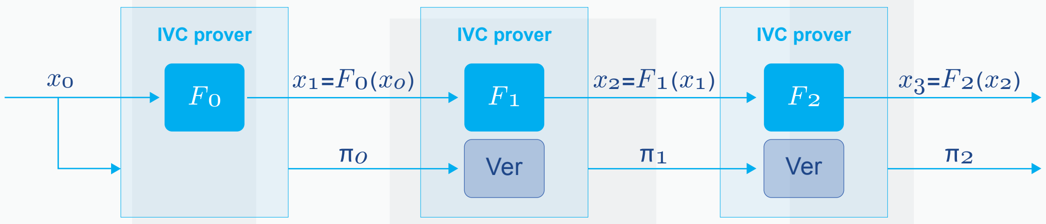

Figure 2: Example of Incrementally–Verifiable Computation.

IVC has plenty of applications, and it is very powerful to construct succinct blockchain systems[2] ,[25]. One could keep a very small proof on-chain, all the participants could verify the validity of the proof to make sure all the historical transactions are valid. The research of IVC is also very active in the academic community.

Proof-Carrying Data. PCD is a generalization of IVC, where the computation is represented with a DAG, not just in a path. The research is very active [5] ,[6] , but the overall performance is still not satisfactory for real applications.

2.4. Accumulators

An accumulator is a one-way membership hash function. It allows users to certify that potential candidates are a member of a certain set with a short and easy-to-verify witness. In some cases, it is also required to conceal the individual members of the set.

One trivial accumulator is Merkle tree, which accumulates all the transactions into the root of the tree. A witness of the membership is the path of its siblings to the root. One could easily recompute the root according to the path and compare the results. Vector commitment does the same as Merkle trees by using a totally different way, it could heavily rely on polynomial commitment. The recent pointproofs[21]system is a new and efficient candidate for vector commitment.

A verkle tree is essentially a combination of vector commitment and Merkle tree, which is first proposed by John Kuszmaul [26]. In a Merkle tree a parent is the hash result of its two children. While in a verkle tree, a parent is the vector commitment of its multiple children. Verkle trees are more space-efficient than Merkle trees, and the witness of membership contains multiple opening proofs of vector commitments, which could be aggregated into single one proof. Due to the efficiency in proof size, Ethereum 2.0 plans to build stateless clients [4] using verkle tree.

3 / On-Chain Capabilities

Privacy and storage are two important issues in blockchain. LatticeX Foundation focuses on the following aspects.

3.1. Privacy

Payment transaction and smart contract are two basic components of blockchain. The privacy requirements and underlying solutions are different.

3.1.1 Payment

It is worth pointing out that the main tokens in Alaya and PlatON are transparent, only the privacy requirements of sub-tokens based on these two networks are considered. Developers could build their applications based on these basic capabilities.

Confidential Transaction is a widely used notion to hide the amount of a transaction. It involves homomorphic encryptions/commitments to conceal the amount, and combines with a range proof to ensure the validity of the transaction. The underlying solutions are slightly different for account model and UTXO model. Since one could customize the mode using smart contract, solutions for both models will be supported.

Anonymous Transaction hides the payer and payee address in the transaction. Anonymity only makes sense when the number of address in the numbers is sufficiently large. We note that in this type of transaction, the amount of the transfer is public. This is useful to build private decentralized exchange based on automated market maker (AMM). In order to provide anonymity, UTXO model is preferable. This is because in an account model, update of the state will leak the address anyway. We encourage the community to propose efficient solutions for account model.

Private Transaction hides the address and amount simultaneously. Several methods are proposed and a relatively complete solution is presented in ZCash based on ZK-SNARK. Although the performance is still an urgent issue, it provides the best privacy in both aspects. Other solutions based on ring signatures are limited to small anonymity set. We will focus on the solution based on ZK-SNARK, since we believe security and privacy is the first priority.

3.1.2 Computation

Smart contract is designed to be Turing machine that could customize all computable functions. It is a generalization of payment, and the difficulty to solve the privacy problems is much higher.

ZK Compiler and ZKVM. Since there may be a variety of applications, it is hard to predetermine which circuit should be used. On the other hand, it is nearly impossible to require all developers to understand the usage and cryptographic background of complicated ZK-SNARK algorithms. This is the main reason to provide ZK compiler and ZKVM to support privacy-preserving applications based on smart contract.

Developers use easy-to-use front-end programming languages to customize their applications,including specify the private inputs. The ZK compiler will automatically transfer this application to suitable circuits or constraints. ZKVM will generate the parameters that used for verification and will be deployed on the blockchain. Meanwhile, ZKVM will automatically generate the zeroknowledge proof when users feed in the private input. This will reduce the bar of use significantly, and several projects [40] ,[13] ,[39] ,[7] ,[29] are dedicated to this area.

Input and Output Privacy. In a simplified way, the execution of smart contract could be abstracted as computing $y = F(x_1, \dots, x_n)$, where x_i's are the inputs, y is the output and F is the application described in the smart contract. A straightforward requirement is to protect the privacy of x_i's and y. If the inputs belong to a single party, one could use a (fully) homomorphic encryption scheme to encrypt the inputs, and outsource the computation to a third node. Combining with verifiable computation algorithms, it will be easy to verify the result. If the inputs belong to multiple parties, one should apply more powerful multi-key homomorphic encryption schemes. Each party encrypts his input with his public key, any third party evaluates the ciphertexts under different keys homomorphically and provides a proof to ensure the correctness.

Function Privacy. The most ambitious goal of private computation is to hide the function itself as well as the inputs and outputs. It will be much more difficult to design the solution.

One candidate framework is proposed in Zexe [3] . It enables parties with access to a blockchain and execute computations offline and subsequently post privacy-preserving, publicly verifiable transactions that attest to the correctness of these off-chain executions. We will integrate Zexe in the first step.

3.2. Storage

The append-only property of blockchain inherently keeps the data increasing as transactions grow. At the time of writing this manuscript, Bitcoin and Ethereum store hundreds of gigabyte amount of data, and the size is still increasing. It takes days on a typical laptop for a full node to download and verify these historical transactions. Although the storage issue is not urgent in PlatON and Alaya, we should prepare and research feasible solutions.

Stateless Client. Full node contains all the historical transactions, the size of the data is too large to be stored in lightweight clients, such as IoT terminals or cell phones. Stateless clients contain only the chain of headers without the execution of any transactions or associated states. The size of the headers is much smaller than the transactions, almost less than 5%. Given a transaction or a state, stateless clients will additionally require a witness that proves the validity. Simple Payment Verification (SPV) provides such a witness, which is essentially merkle tree branches, to convince clients without storing all the transactions. The support of stateless clients in PlatON and Alaya will significantly reduce the threshold to get involved in the network. Edge-side devices can also access the network and conduct business transactions, which can greatly expand the network's reach.

Stateless Chain. A more ambitious goal is to upgrade PlatON and Alaya into stateless chains. In a stateless blockchain, only succinct information (for instance, a constant-size proof) is stored on-chain instead of state information. Mina [2] is the first of this kind blockchain to achieve statelessness. It involves in using recursive zero-knowledge proofs to prove the validity of states and only keep constant-size data on-chain. In order to solve the data availability issue, the states and transactions need a place to store anyway, we introduce storage services in the next section. It is worth pointing out that upgrading to stateless chain will significantly change the current layer-1 architecture of PlatON and Alaya, thus we leave it as a long-time planning.

4 / Off-Chain Services

As discussed in previous sections, it is inevitable to provide computation and storage services to address the privacy and storage issues. The off-chain services are built upon layer 2 in PlatON 2.0.

4.1. Computation Service

In the layer 2 of PlatON 2.0, institutes or nodes with enormous amount of computation power could join the network to provide computation services as follows.

Private zk-Rollup. As described in Section 3.1 , cryptographic tools are utilized to solve privacy problems. Due to the heavy cost of cryptographic operations, the performance will be reduced significantly. Zk-Rollup is widely discussed in the Ethereum community to improve scalability. In a nutshell, an off-chain service provider (or node) packs a batch of transactions, and provides a single proof to ensure the validity of all the transactions. The on-chain verifier (probably a smart contract) verifies this single proof to validate all the batch of transactions.

We will borrow this idea to improve the scalability of our systems after the privacy-preserving properties are supported. The motivation here is slightly different from Ethereum. The performance of original PlatON/Alaya is well enough for most applications (not considering privacy), and the scalability issue is incurred by introducing private payments and computations. So we call it private zk-Rollup.

Eth-PlatON Node. It is well known that the Ethereum community is one of the most creative communities. The computation service in PlatON network also aims to combine with Ethereum to solve the scalability issues. For instance, these node could provide zk-Rollup services for Ethereum,and users could enjoy the benefit from PlatON (low transaction fee) and Ethereum (rich applications). In more detail, users who have deployed applications in Ethereum could pay a PlatON node with LAT to pack all the Ethereum transactions. This node batches all the transactions and generates a proof for the Ethereum network. Thus the users only need to pay a tiny amount of ETH fees and LAT fees.

Secure Outsourcing. The off-chain computation service also enables secure outsourcing. PlatON will provide a decentralized outsourcing and fair-exchange market place. Clients with low computation power could publish a computing job, where the computed functions, fee and other related information are specified in a smart contract. In order to protect the privacy of input data, the client encrypts all the input with a fully homomorphic encryption scheme and sends the ciphertext to a computation node who picks up the job. This node evaluates the computation in an encrypted form and provides a proof to convince the smart contract that he did the job correctly. The smart contract verifies the proof and sends the tokens to this node if the verification passes.

With a single node as the first step to build computation service in the layer 2 of PlatON, it will finally form a computation marketplace for whom want to sell their computation power in a decentralized manner. Besides providing internal services for PlatON and Alaya, these off-chain services could also be extended for other systems and networks.

4.2. Storage Service

Although there are several methods to reduce the on-chain storage size, there has to be a place to store all the transactions anyway. In the lay 2 of PlatON, anyone who has sufficient disk space could provide storage services.

Data availability is the core problem we need to solve in storage services. Since historical transactions are stored off-chain, one has to make sure that there existing at least one node stores these data. New transactions initiated by users may change the state related to historical transactions. For example, the system has to ensure you could spend the tokens that received 10 years ago. Several ways may could be applied here, the first one is proof of custody, which is considered in Ethereum 2.0 to solve data availability issues. Although the situations are slightly different from Eth 2.0 and PlatON, we encourage deve-opers to intensively study the solutions. Other possible ways are integrate proofs of storage and proofs of replications algorithms into our systems. All those solutions are heavily relied on cryptographic tools.

Storage services providers will finally work in a distributed manner. Access control of the data will be a challenge in this situation, especially when sensitive data is stored in multiple providers. Proxy re-encryption (PRE) enables users to upload encrypted data into these providers, and could share with other parties only under the owner’s permission.

5 / Applications

In this section, we generally describe several applications based on privacy-preserving capabilities.

5.1. Fair Exchange

Fair exchange enables a seller and a buyer to exchange data securely. Security means that the buyer pays to the seller if and only if the buyer gets the digital goods. Theoretical result shows that it is impossible to exchange fairly without a trusted third party. However, it is possible to take the public blockchain system as a trusted third party to design fair exchange protocols.

In general, an efficient protocol for fair exchange of digital goods uses smart contracts. A fair exchange protocol allows a sender to sell a digital commodity x for a fixed price p to a buyer. A typical solution of fair exchange is "Zero Knowledge Contingent Payment" (ZKCP) [33], [8], [16], ZKCP technique enables fair exchange to be achieved by using blockchain, where Bitcoins are released if and only if some knowledge is disclosed by the payee. In particular, ZKCP uses zero knowledge proof algorithms together with a hash-locked transaction to make sure the revealed data in the released hashlock is the data that the payer need. Using the progress in ZK-SNARK [32], we could even exchange a large amount of data which satisfies properties featured by deep learning models.

5.2. Anonymous DAO

Decentralized Autonomous Organization (DAO) is an organization that was designed to be automated and decentralized. It acted as a form of venture capital fund, based on open-source code and without a typical management structure or board of directors. To be fully decentralized, the DAO was unaffiliated with any particular nation-state, though it made use of the public blockchain network. Anonymous DAO aims to improve participants' privacy by using cryptographic privacypreserving techniques like zk-snarks.

People can create a proposal by proving the validity of their membership within the group. Also DAO needs to prevent spam proposals by requiring any proposal to be uniquely created in an epoch. People can make votes to any proposal by proving their memberships without disclosing their identities, and proving the freshness of their votes. Technically speaking, one could use ZK-SNARKs to prove the following and achieve the above requirements: a) Membership of a Merkle tree. b) The correctness of a nullifier, where a nullifier is an unforgettable identifier of an item (a vote, a membership, etc).

5.3. Private Decentralized Exchange

A private DEX is a decentralized exchange that protects the identity and the amount of the traders. Current DEXs mostly focus on automated market maker (AMM) modes. An automated liquidity protocol powered by a constant product formula and implemented in a system of non-upgradeable smart contracts on the blockchain. It obviates the need for trusted intermediaries, prioritizing decentralization, censorship resistance, and security.

Anyone can become a liquidity provider (LP) for a pool by depositing an equivalent value of each underlying token in return for pool tokens. These tokens track pro-rata LP shares of the total reserves, and can be redeemed for the underlying assets at any time. Pairs act as automated market makers, standing ready to accept one token for the other as long as the "constant product" formula is preserved. This formula, most simply expressed as $x * y = k$, states that trades must not change the product (k) of a pair's reserve balances (x and y). Because k remains unchanged from the reference frame of a trade, it is often referred to as the invariant.

Private payment could be used to construct private DEXs. Anonymity could be held by applying anonymous transaction based on ZK-SNARKs. However, in the AMM model, it is difficult to protect the transfer amount. This is because the changed values Δx and Δy have to satisfy the constraint $(x - \Delta x) * (y + \Delta y) = k$, and these two values should be public for all the participants. It is still an open problem to provide anonymity and confidentiality for DEXs.

5.4. Sealed Bidding

Sealed Bidding provides a privacy-preserving manner for the auction where no bidder learns any information about the other bids. The bidders are encouraged to bid according to their monetary valuation of the asset. On the other hand, the existence of any collusion between the auctioneer and a malicious bidder can break the advantage. To prevent the conflict between protecting the privacy of the bids and trusting the auctioneer to individually determine the winner, cryptographic protocols can be utilized to accomplish the publicly verifiable correctness without sacrificing the privacy of the bids.

A typical cryptographic sealed-bid could involve the auctioneer, different bidders and the auction smart contract to interact with each other, where they use primitives like homomorphic commitment schemes, zero knowledge proofs for interval memberships, etc. The auctioneer initially deploys the auction contract on the blockchain, with some parameters to be configured, including amount of initial deposit of bidders, time intervals, maximum number of bidders, etc. During the bidding process, each bidder submits a commitment of his bid to the auction contract, and also encryption of the bid witness and randomness to the auctioneer. The auctioneer orders the bids according to the correctness to determine the winning bid, the associated account address and commitment,and proves the correctness of the winner simultaneously .

5.5. Private NFT

Private non-fungible Tokens (NFT) enable verifiable representation of unique items and events, such that ownership and transactions are private by default. Private NFT can be specialized with different ZK-SNARK circuits based on different scenarios. For instance, an invoice which dictates that Bob will pay Alice $1000 in 90 days, can be minted as an NFT that represents a claim on the future revenue. The owner of the invoice Alice can transfer this NFT to Cindy by making on-chain trading of this asset.

A simple NFT registry based on a smart contract that lets anyone mint an NFT by providing a ZK-SNARK proof that validates the NFT completely off-chain. With that, we can achieve that the private information in the off-chain document stays private, but all NFTs have verifiable attributes.

5.6. ZK-Game

ZK-Game is essentially a type of “game with incomplete information”, where players do not possess full information about their opponents and the environments. Some players possess private information, a fact that the others should take into account when forming expectations about how those players will behave. For instance, Poker is a typical game with incomplete information, where a player never knows the cards in his opponents. Game with incomplete information allows people to explore a richer and more dramatic strategy. Information asymmetry enables deceive, conditional coordinate, and complex social dynamics.

Using ZKP tools like ZK-SNARKs in building games with incomplete information is quite crucial. For instance, ZK-SNARKs makes it possible to build and verify claims like “I moved my horse from a secret location A to a secret location B. I will not tell you any information of location A and location B, but this proof proves that the movement from A to B is indeed valid”.

Dark Forest is built on ZKP as a game with incomplete information in Ethereum. One of the core ideas behind Dark Forest is the “cryptographic fog of war” protected by ZK-SNARKs. We encourage developers to create interesting games as Dark Forest in our systems.

6 / Distributed Key Management Services

Blockchain provides the underlying core technology of decentralized networks. In all these blockchain systems, decentralization essentially means that each individual could control his own asset by means of controlling the private key (or signing key) of digital signatures. A transaction (transfer of assets) is valid as long as the underlying signature passes the verification[2]. However, controllingthe private keys individually may incur many problems in practice.

- **Usability.** Private/signing key differs from password. In most of our daily applications, people get used to keeping in mind the passwords with specific meanings. However, private keys are random strings with 32 bytes length. It is barely possible to remember all these meaningless strings when participants initiate transactions. Several methods are proposed to manage private keys, but all of them have some drawbacks.

– *Mnemonic words.* Mnemonic words are generated from standard algorithm, which takes a private key as input. These words have specific meaning and could be converted back to a private key. Although it is easy to remember one single word, there are always more than 10 words to represent one private key. Therefore, Mnemonic words are commonly used as backups of private keys and stored in physical materials. For instance, write them down in a paper, and put it into a safe deposit box.

– *Keystore.* In most decentralized wallets, passwords are used to protect private keys.Private keys are encrypted using symmetric ciphers (e.g., AES) by taking the password as a secret key. The resulting ciphertext forms the keystore file, which is stored locally in the device. Keystore file significant simplifies the usage of private keys, in which the usage is essentially the same as centralized Apps. However, when users change devices, they have to import the keystore files, this will increase the complexity when changing devices.

– *Hardware Wallet.* Users who have higher security requirements may prefer to use hardware wallet. The private keys are generated and used in a safe and trusted area in the hardware wallet. Private keys will never leave this area in the whole life circle. The underlying security is essentially provided by the hardware, which controls the access of private keys. Thus, users have to carry a customized device or card to deal with their asset, and these devices are commonly suitable for laptops, not for mobile phones.

[2] States or other related conditions should be checked as well.

- **Security.** The one who has the private key could totally control the asset, the security of private keys directly affect the security of the asset. Most events such as exchange security incident and asset loss are caused by mishaps of private key management. Loss of mnemonic words, keystore files and hardware wallets happens frequently, and the lack of retrieval mechanism prevents users finding the private key back. Once users lose private keys, loss them forever.

  Although many approaches are proposed to protect private keys, we should pay more attention to one crucial attack surface. That is, the private key appears in the memory in plain when using it to sign a transaction. Despite of a very short time slot, the attackers could easily steal the whole private key and then all the assets without sophisticated precautions.

- **Lack of Approval Procedure.** In the traditional financial management procedure, the approval of large fund transfers is very complicated, and each enterprise may have specific polices. For example, the transfers should be approved by the finance specialist, CFO and CEO sequentially, before they are initiated. However, these polices could not apply to blockchain directly. This is because the private-key owner could ignore all the polices, and initiate transfers as he wants.

  One commonly used method to solve the above problem is multi-signature. Roughly speaking, this approach relies on smart contract (or script in Bitcoin). The smart contract states that “this transaction is valid as long as three collected signatures are valid”, and execution of the transfer will exactly follow the constraints as stated. Theoretically, smart contract could specify arbitrary polices for different purpose, but it still has some drawbacks.

– Smart contracts will be run by every node in a repetition execution way. Thus the policies stated in smart contract are publicly known to all participants. This is not desirable for enterprise customers, since the policy may leak sensitive information from the company.

  Another serious issue is that the policy is also public to hackers, who may have interests in attacking the company. Contract security has been one of the important problems plaguing the entire blockchain community.

– For custody providers, they always need to maintain hundreds of blockchain assets and tokens. These heterogeneous blockchain may design different smart contract systems or even do not support smart contract. The cost of developing policy-related smart contract will be extremely high for these custody providers. The developers has to learn different programming languages, complete and sophisticated security test of these resulting contracts should be taken into account.

It is worth pointing out that the requirement for key management is much urgent for enterprise users. Individual users may prefer to use centralized wallets for high-frequency trading, and actually do not possess the private key.

In order to solve the above problems, we aim to utilize cryptographic protocol to provide distributed key management systems. More specifically, the underlying multi-party computation (MPC) protocol is threshold signatures. Utilization these tools and protocols are arising in the blockchian community, but still lack of commercial use and standard.

6.1. Threshold Signature

Threshold cryptography enables a set of parties to carry out cryptographic operations, without any single party holding the secret information. Threshold signature is a subfield of threshold cryptography, which focuses on distributed signing. In cryptography community, one may design new signature schemes that are suitable to convert into the threshold version. But this is not desirable in blockchian community, because specific and standard signature schemes, such as ECDSA/EdDSA/BLS, are already adopted. It is extremely difficult to replace them with new ones. Therefore, this direction aims to design efficient MPC protocols for existing signature schemes.

In general, threshold signature comprises the following two main components.

- **Distributed Key Generation.** In a traditional key generation procedure, one chooses a uniformly random private key x , and generates a public key locally. In a distributed key generation procedure, n participants choose uniformly random shares $x_1, ..., x_n$ locally, and run a MPC protocol to generate the public key without collecting all these shares together. The resulting private key is $x = x_1 + \cdots + x_n$ and does not appear in the procedure.

- **Distributed Signing.** A signing algorithm takes as input the message and the private key, and outputs a digital signature. In a distributed version, each participant holds xi and runs a MPC protocol to compute the signing process without collecting x_1, ..., x_n together. The resulting signature is essentially the same as the local version.

Plenty of threshold signature schemes have been proposed [30], [14], [9] , [18], [31] , [15], [10], [11], [36] . LatticeX Foundation encourages cryptography experts to implement these protocols and make them open-source.

6.2. Framework and Architecture

Framework. The aim of distributed key management services based on threshold signature is to provide a unified framework to guarantee security, usability and flexibility. The basic framework is illustrated as follows in Figure 3 .

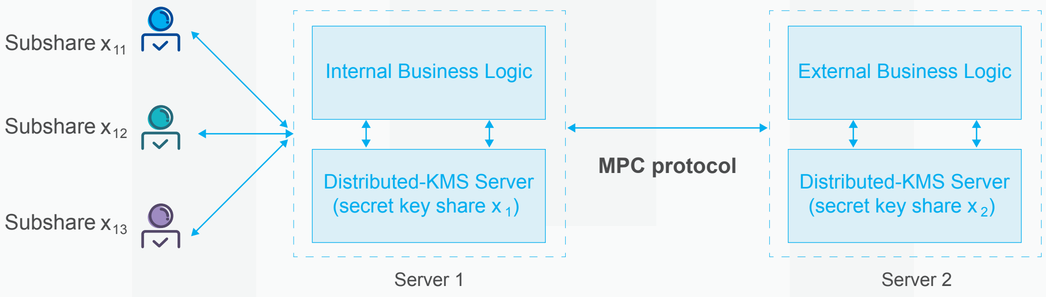

Figure 3: Framework of Distributed Key Management Services.

The core component is a distributed KMS server, which could be deployed in different environments. Although only two servers are depicted in Figure 3 , one could easily extend them to multiple servers. The servers interact with each other to run threshold signature protocols, and provide standard APIs for developers to customize their business logic, including the policies.

Two levels of policies are considered in this framework. The private keys are shared between severs according to some policies which we called Level-1 policies. The server himself could reshare the previous shares into subshares according to other polices, which we call Level-2 policies. These subshares are always distributed to individual users, which is essentially generated by the (n, t)-Shamir Secret Sharing scheme.

The basic workflow is stated as follows.

- When a group of participants require to create a jointly managed account, the two servers call the distributed key generation procedure and obtain x_1 and x_2 , respectively. Server 1 reshares $x_1$according to Level-2 policies ($e.g., 3\text{-}out\text{-}of\text{-}3\ policy$), gets x_{11} , x_{12} , x_{13} and sends them to the individual users respectively. Server 1 will delete all the shares at the end of this procedure.

- A Large fund transfer could be initiated by the system or one participant in the group. If all the participants (or other conditions satisfy the Level-2 policy) approve the transfer, they will send the subshares to Server 1. Server 1 confirms the approval, collects the subshares, reconstructs the share x_1 and runs the threshold signature protocol with Server 2 to complete the transfer.

This framework has many advantages covering security, usability and flexibility, which are described as follows.

- **Security.** Inheriting from threshold signature, the private key never appears in the life cycle. The shares (of Server 2) could be managed by traditional KMS (see Figure [4]). For individual users, the subshare could be stored in mobile phone. It will not affect the security of the whole system even they lost the subshares. Additionally, traditional authentication method, such as two-factor authentication, could be applied to enhance the security. One could even refresh the subshares in a fixed period of time or manually.

- **Usability.** One important advantage of this framework is that the end-side users do not need to approve the transfer on-line simultaneously. They simply send the subshares independently and the interactive protocol is actually handled among the servers.

- **Flexibility.** The users could set customized Level-1 or Level-2 policies according to their business logic. It is also very flexible to design retrieval mechanisms by combine policy levels. For instance, one could set Level-1 policy be (2, 2) and Level-2 policy be (3, 1), then the retrieval service is provided by Server 1. Or one could set a (3, 1) Level-1 policy, then the retrieval procedure is provided by all the three servers.

Architecture. The core component of the framework is the DKMS server. The architecture of the server is depicted in Figure [4] .

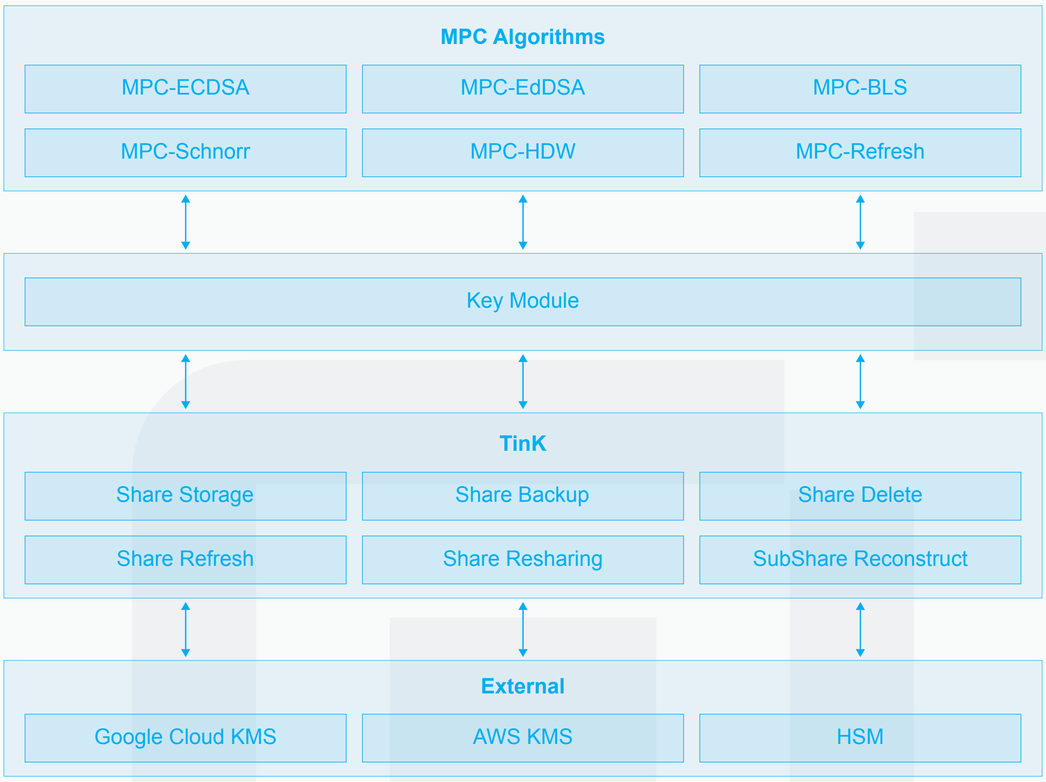

Figure 4: The Architecture of DKMS Server.

At the heart of the server are various threshold signature protocols that supporting commonly used digital signature schemes, including ECDSA, EdDSA, BLS and Schnorr. Note that the difficulty is to design efficient threshold signature protocols for ECDSA and EdDSA. Other MPC protocols to support hierarchical deterministic wallet and share refreshing are also very useful in practice.

In the early version, a middleware called Key Module is designed to connect MPC protocols and Tink [20] , which is an industrial-grade framework developed by Google. In this version, the mature tink framework is used to help manage the shares. A long-term version will integrate these protocols into tink.

Tink is a multi-language, cross-platform, open source library that provides cryptographic APIs that are secure, easy to use correctly, and hard to misuse. It reduces common cryptography pitfalls with user-centered design, careful implementation and code reviews, and extensive testing. At Google, Tink is one of the standard cryptography libraries, and has been deployed in hundreds of products and systems. Tink was born out of our extensive experience working with Google's product teams, fixing weaknesses in implementations, and providing simple APIs that can be used safely without needing a cryptography background.

Tink also supports external key management service, such as Amazon KMS, Googl Cloud KMS, Android Keystore, and iOS Keychain. For commercial usage, these standard and widely accepted KMS systems are preferred to manage shares.

7 / Roadmap

The preliminary roadmap is given in Figure 5.

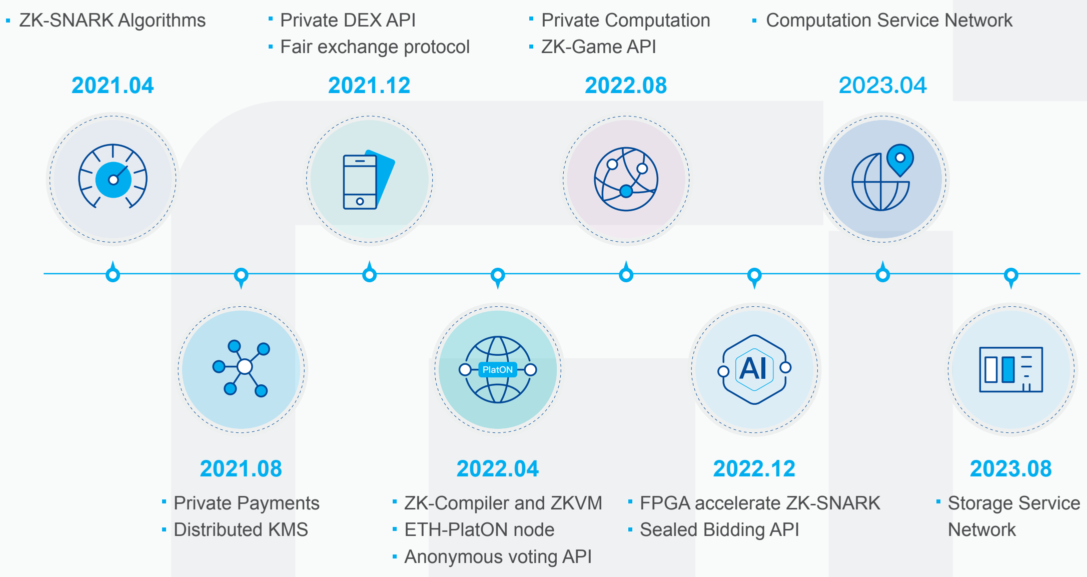

Figure 5: The Roadmap.

2021 Q2

- Support basic ZK-SNARKs algorithms, including Plonk.
- Support basic primitives, such as zk-friendly hashes et al.

2021 Q3

- Support private payments, including confidential transaction, anonymous transaction and private transaction.
- Publish open-source distributed key management services.

2021 Q4

- Publish private DEX API, protect the anonymity of transactions.
- Release fair exchange protocols.

2022 Q2

- Release ZK-Compiler and ZKVM wicha are adapted to Plonk.
- Build ETH-PlatON node that connect Ethereum and PlatON.
- Release Anonymous voting APIs.

2022 Q3

- Publish private computation framework based Plonk.
- Release ZK-Game APIs.

2022 Q4

- Release FPGA accelerate source code for Plonk.
- Release sealed bidding APIs.

2023 Q2

- Release computation service network.

2023 Q3

- Release storage service network.

REFERENCES

[1] Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. In: FSE (2020)

[2] Bonneau, J., Meckler, I., Rao, V., Shapiro, E.: Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352 (2020), https://eprint.iacr.org/2020/352

[3] Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: Zexe: Enabling decentralized private computation. In: IEEE Symposium on Security and Privacy (2020)

[4] Buterin, V.: Verkle trees (2021), https://vitalik.ca/general/2021/06/18/verkle.html

[5] B¨unz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. In: Advances in Cryptology - CRYPTO (2020)

[6] B¨unz, B., Chiesa, A., Mishra, P., Spooner, N.: Proof-carrying data from accumulation schemes. In: TCC (2020)

[7] Cairo-Lang: https://github.com/starkware-libs/cairo-lang

[8] Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L.: Zero-knowledge contingent pay□ments revisited: Attacks and payments for services. In: ACM CCS (2017)

[9] Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Advances in Cryptology - CRYPTO (2019)

[10] Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DSA. In: Public-Key Cryptography - PKC (2020)

[11] Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts, proactivity and adaptive security. eprint/2021/291 (2021)

[12] Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. In: Advances in Cryptology - EUROCRYPT (2020)

[13] Circom: https://github.com/iden3/circom

[14] Doerner, J., Kondi, Y., Lee, E., abhi shelat: Secure two-party threshold ECDSA from ECDSA assumptions. In: IEEE Symposium on Security and Privacy (2018)

[15] Doerner, J., Kondi, Y., Lee, E., abhi shelat: Threshold ECDSA from ECDSA assumptions: The multiparty case. In: IEEE Symposium on Security and Privacy (2019)

[16] Fuchsbauer, G.: Wi is not enough: Zero-knowledge contingent (service) payments revisited. In: ACM CCS (2019)

[17] Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report2019/953 (2019), https://eprint.iacr.org/2019/953

[18] Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: ACM CCS (2018)

[19] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: STOC (2008)

[20] Google: Tink. https://github.com/google/tink

[21] Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: Aggregating proofs for multiple vector commitments. In: ACM CCS (2020)

[22] Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. In: Usenix Security (2021)

[23] Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology - EUROCRYPT (2016)

[24] Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Advances in Cryptology - ASIACRYPT (2010)

[25] KKattis, A., Bonneau, J.: Proof of necessary work: Succinct state verification with fairness guarantees. Cryptology ePrint Archive, Report 2020/190 (2020), https://eprint.iacr.org/2020/190

[26] Kuszmaul, J.: Verkle trees (2018), https://math.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf

[27] LatticeX Foundation: Lumino: Light up the Evolving Road. https://lumino.latticex.foundation/home (2021)

[28] LatticeX Foundation: PlatON 2.0: Decentralized Privacy-Preserving AI Network (2021)

[29] Leo: https://github.com/AleoHQ/leo

[30] Lindell, Y.: Fast secure two-party ECDSA signing. In: Advances in Cryptology - CRYPTO (2017)

[31] Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: ACM CCS (2018)

[32] Liu, T., Xie, X., Zhang, Y.: zkcnn: Zero knowledge proofs for convolutional neural net□work predictions and accuracy. Cryptology ePrint Archive, Report 2021/673 (2021), https://eprint.iacr.org/2021/673

[33] Maxwell, G.: Zero knowledge contingent payment (2015), https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment

[34] Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: TCC (2008)

[35] Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Advances in Cryptology - CRYPTO (2019)

[36] Yuen, T.H., Cui, H., Xie, X.: Compact zero-knowledge proofs for threshold ECDSA with trustless setup. In: Public-Key Cryptography - PKC (2021)

[37] Zhang, J., Liu, T., Wang, W., Zhang, Y., Song, D., Xie, X., Zhang, Y.: Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. Cryptology ePrint Archive, Report 2020/1247 (2020), https://eprint.iacr.org/2020/1247

[38] Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: IEEE Symposium on Security and Privacy (2020)

[39] Zinc: https://github.com/matter-labs/zinc

[40] ZoKrates: https://github.com/Zokrates/ZoKrates