# Decentralisation as a service

Whitepaper

**fern.network**

Developed by Applied Blockchain
**appliedblockchain.com**

## Table of Contents

## MOTIVATION

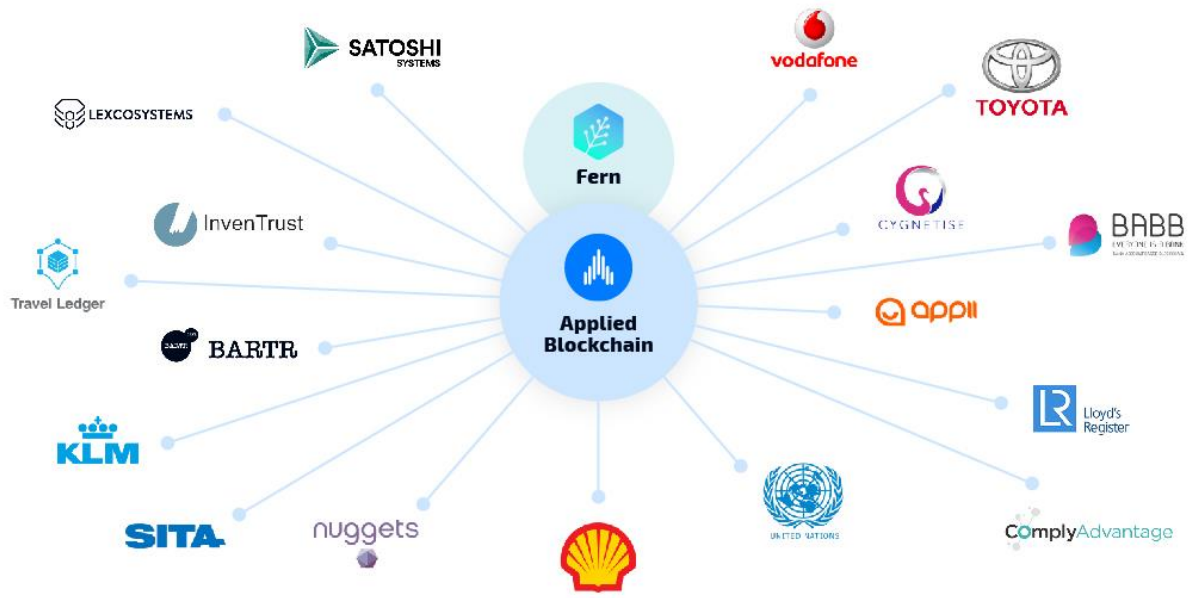
The promise of blockchain involves decentralization of computing and immutable trust in distributed networks and storage. The vision of the original public blockchains is numerous cross-industry applications co-existing on a single network. The reality today is that most of the estimated 24,000 blockchain proof of concepts and applications deployed in the last 2 years are in dedicated private networks or networks with a limited number of nodes. While these networks provide the ability to control access to data and greater transaction throughput and scalability, they do also have their drawbacks, limiting the ability for blockchain applications to be fully trusted and adopted.

These blockchain-based networks are not able to achieve the security promised by blockchain technology. When node identity, and their physical owner, is known to all, a 51% attack becomes more likely. In this case, nodes and the people and organisations controlling them, can collude to form a majority and act maliciously to defraud the ledger. The more limited in size the network, the more chance for collusion. Many promising but nascent blockchain applications are not secure for this reason. On the other hand, networks depending on fully open public blockchains for transaction verification suffer from high transaction costs and low throughput.

As a result, there is a paradox in the growth of blockchain networks and their applications. Limited node networks lack trust, yet large-scaled networks are too unwieldy to be useful. In a practical sense, there is no easy way to deploy a blockchain application that is business-ready, compliant, scalable yet secure from potential collusion.

To solve this issue, nodes are required to demonstrate two distinct properties: (1) identity, and (2) anonymity. Identity in that they are known and therefore known to be compliant with business and regulatory requirements, and anonymity in that they can be provisioned without being compromised via identity. Until now, these two qualities were incompatible, a node is was either identified or anonymous.

Conceived by a development team at Applied Blockchain, with combined 3+ years' experience developing blockchain applications for some of the world's largest businesses and most promising start-ups, the Fern network provides decentralization and trust as a service, connecting a community of identifiable yet anonymous node and oracle services.



Applied Blockchain's clients include some of the world's largest organisations in a number of industries including energy, aviation, automotive, shipping, supply chain, financial services and telecommunications. The client list includes Shell (also an investor), Vodafone, SITA (owned by the airline industry), KLM, UN, Toyota, Lloyds Register (global registry of ships), and a range of blockchain start-ups deploying real world production applications. Fern has already on-boarded nine members of the community, these include decentralised applications and networks, as well as oracles.

## VALUE PROPOSITION

The reason so many enterprises are investing in blockchain is that they see potential efficiencies in their respective markets. These generally come in 3 stages: multi-party process and data efficiencies (reduced reconciliation, disputes, intermediaries), asset registration and ownership transfer efficiencies, and value transfer (i.e. payment, tokens) efficiencies.
As they process along this path, two things occur:

1. They gain more benefit (e.g. if the entire business case for a complete industry blockchain solution is worth $100m, the first stage will return 10% through efficiencies)
2. The value on the ledger starts to increase: data, then assets, then tokens of value; and therefore the associated risk if fraud were to occur grows.


So, they need to progress along this path in order to realise efficiencies and benefits of their business case, BUT the cost of a breach or malicious activity goes up substantially, and they must now invest in a solution to guard against this. They must decentralise their blockchain network to reduce likelihood of fraud or collusion to undermine the ledger and the value recorded on it. This value through true decentralisation is the value brought by Fern.

To illustrate the value of true decentralisation for enterprises, and protection from a 51% collusion attack, one only need to look at the Ethereum Classic public network[21]. Research shows that conducting a 51% attack on the Ethereum Classic network at the time of writing would cost just $70m to release $1Bn in profit[1]. The low cost required to conduct a 51% collusion attack stems from the lack of decentralisation, and undermines the security, and therefore the value of that network.

## INTRODUCING FERN

Fern provides decentralization as a service. Our goal is to allow any decentralized application network to add security and trust by providing access to a community of identifiable, yet anonymous, independent nodes and oracles.

Decentralized application networks that require additional, independent nodes and oracles can request them from the Fern network. These deliver additional decentralization and trust providing users of the applications further protection from collusion, fraud and malicious attacks.

Fern establishes a community of third-party nodes and oracles by recording their membership keys in both identified and anonymous form. The identified keys reveal a public profile that can be verified for assurance

[1] Note: in a public network such as ETC 51% limited protection is offered through Proof of Work, while in a permissioned network some protection is offered through a limitation on the number of nodes. Further protection is required to abstract their identities to avoid the threat of collusion and bribery.

and compliance; the anonymous keys are used to sign activity on the blockchain. By separating the two keys, a Fern community member can separate physical identity from their blockchain activity. All members of the Fern community belong to a cryptographic ring used to prove that the anonymous blockchain activity is associated with one of the public identities in the community.

Fern also manages: membership, verification, reputation, transactions, consensus, governance, permission, encryption, metering, reward, and oracle and other external service connectivity.

The benefits of the Fern approach are:

- Secure – Systematically add nodes and oracles as needed to ensure a high level of security in any decentralized application network
- Innovative – dual-security method to separate identity vs anonymity, and secured by Fern protocol
- Flexible – Fern integrates with the most widely used smart contract based blockchain software including Ethereum, Hyperledger Fabric, NEO, Stellar, EOS, among others
- Scalable – Fern can scale to thousands of nodes and oracles as needed for each decentralized application network

Fern was conceived from a very real need to deploy decentralized applications in a more secure yet compliant environment. Applied Blockchain has already built and deployed numerous decentralized application networks, some as live production environments, that all require and will benefit from decentralization as a service. Examples include:

- Invoicing, invoice financing, invoice trading (Tallysticks)
- Immutable education and career history for efficient recruitment (APPII)
- IP registration network (InvenTrust)
- Consumer data privacy (Nuggets)
- Banking for the underbanked (BABB)
- Telecoms wholesale market and protocol (BARTR)
- Commodity Repo Financing (Minerva)
- Legal smart contracts (Lexcosystems)
- Company administration (Cygnetise)
- Regulated stable coin infrastructure (Supermoney)
- International drone registry (SITA)
- Travel payments network (TravelLedger)
- Aircraft component tracking (KLM)
- Registry of ships (Lloyds Register)
- Energy trading and supply chain (Royal Dutch Shell)
- Vehicle financing (Toyota)
- Decentralized data privacy (Italian telco)
- Group finance settlement (Vodafone)
- Aid distribution (UN WFP)

Technical Whitepaper v1.0.64

SYNOPSYS

Current blockchain architectures are relatively successful at supporting cryptocurrency networks, crypto token issuance, and smart contract applications.

Anonymous blockchain nodes found in public networks provide decentralised control and security yet are impractical for many commercial scenarios where varying levels of identity and compliance are required. Identifiable blockchain nodes and oracles, typically found in private networks, enable compliance, yet are known to each other, and therefore susceptible to collusion and fraud.

This paper solves this dichotomy using pools of identified actors with random and anonymous subsets supporting distributed application networks. This enables secure decentralizing, connecting and scaling of private, layer 2 and some public networks.

## 1. BACKGROUND

We begin by presenting a number of common attack vectors, scalability considerations, and limitations of current blockchain configurations.

Vitalik Buterin, founder of Ethereum, describes three types of decentralisation[19]:

*Architectural (de)centralisation — how many physical computers is a system made up of? How many of those computers can it tolerate breaking down at any single time?*

*Political (de)centralisation — how many individuals or organizations ultimately control the computers that the system is made up of?*

*Logical (de)centralisation — does the interface and data structures that the system presents and maintains look more like a single monolithic object, or an amorphous swarm? One simple heuristic is: if you cut the system in half, including both providers and users, will both halves continue to fully operate as independent units?*

The first, architectural, adds redundancy, and the third, logical, adds a common language, but these are not specific or unique to blockchains. The second, political, is the decentralisation type that adds trust and security to blockchains through decentralising *control* of the ledger.

Different controlling parties collectively secure and control the ledger. It is the parties controlling the blockchain network nodes, rather than the nodes themselves that provide decentralised blockchain security.

### 1.1. Sybil and Collusion Attack Threats

A Sybil attack involves a party assuming multiple identities in a peer to peer system in order to conduct an attack. A blockchain Sybil attack involves a party assuming multiple anonymous node identities in order to undermine the blockchain consensus protocol.

The blockchain can be undermined by malicious nodes ignoring, blocking, injecting or reordering transactions as presented in a new block. The more nodes a party controls in a network, the greater the attack surface. A party controlling a consensus majority (51% in proof of work, 66% in many byzantine fault tolerant protocols) could cause further damage by presenting and approving blocks of transactions that may not have otherwise been approved by independent nodes. Potential attacks include double spend (transfer of token / asset ownership) or state change, followed by a majority-lead fork to restore order after the benefits of the attack have been realized. A party controlling a network consensus majority could even force a fork with historical changes to the ledger, although, depending on transparency of the ledger, this is likely to be detected and reported after the event.

### 1.2. Attack Strategies

The most common Sybil and collusion attack strategies involve:

**Adding**: Malicious actor adds more nodes to a network to gain a majority. Proof of work blockchains protect against this by requiring each node to conduct expensive mining, proof of stake blockchains attach an arbitrary minimum stake to each new node, and restricted node-set (permissioned, private, even some public) blockchain networks simply restrict the number of nodes that can be added to the network through governance.

**Bribing**: Malicious actor bribes or influences the majority of nodes in a network to vote in favor of a new block containing the fraudulent transactions. Fully open public blockchains protect against this through node anonymity. If you don't know who controls the other nodes in a network, you cannot bribe or corrupt them. Networks where node parties are identified are exposed to this collusion attack vector.

### 1.3. Anonymity vs Scalability

Fully public blockchain networks, such as the public Ethereum network, provide a decentralised environment where all participants, nodes and transacting parties are anonymous.

Anonymity of node providers protects the network against collusion-related attack vectors including the Sybil attack.

However, fully public, fully open and fully anonymous networks are notoriously difficult to scale, as they must cope with consensus across a boundless number of nodes.

*"Present-day blockchain architectures all suffer from a number of issues not least practical means of extensibility and scalability. "*
- Dr. Gavin Wood, Founder, Ethereum & Parity [1]

*"Scalability is probably problem number one [...] There's a graveyard of systems that claim to solve the scalability problem but don't. It's a very significant and hard challenge. These are just known facts."*
- Vitalik Buterin, Ethereum Founder[3]

### 1.4. Future of Scalability

Wood and Buterin propose differing approaches to solve the scalability problem in the public Ethereum network, the former through a mesh of interchained networks (Polkadot[1][2]), and the latter by extending the main Ethereum network through a sharding universe and/or Plasma chains[11].

Ultimately both propose delegation, and offloading of scalability requirements to a set of sub networks (also known as tier 2 scaling solutions) connected to a main public network, and, more importantly, both suggest that in the short to near-term a single, a fully open, fully decentralised, fully scalable main public network is not realistic.

Similarly, the Cosmos project, conceived by Ethan Buchman and Jae Kwon proposes a mesh of blockchain networks exchanging assets and connected through a public hub network powered by the Tendermint practical byzantine fault tolerant (PBFT) consensus algorithm. [4]

Plasma [11], a project conceived by Joseph Poon and Vitalik Buterin, proposes a tree hierarchy of blockchains where Merkle proofs are presented to each parent chain.

### 1.5. Limited Node Sets

Numerous platforms claim to solve the public blockchain scalability challenge by restricting the size and population of the node set (e.g. EOS[19]), leading to a simpler and better performing consensus system (increased number of transactions/ second).

A major share of blockchain applications developed and deployed at the present time are practically deployed in limited blockchain network environments, also known as testnets, private, consortium, permissioned, or public blockchains, all with the common features that the nodes set is limited in number and identified.

A number of popular platforms support this type of configuration: Ethereum, Quorum, Hyperledger Fabric, Corda. Some are designed for execution only in a closed, restricted network environment, and some relax the absolute node / ledger replication construct, as well as other core blockchain design principles and assumptions, in order to provide enhanced data privacy, scalability and performance. Hardware secure private

blockchain execution environments (e.g. SGX) provide a further restricted environment in this regard.

Tier 2 scaling solutions such as Polkadot with Parachains, Cosmos with Zones, and Plasma all assume a limited node-set in the second tier. Ancillary distributed services for off-chain data storage and private computation also often depend on limited node-sets.

The limited node-sets provide semi-trusted environments where security and integrity of transaction history are enhanced through the distributed network of nodes, where each node is controlled independently by a different party. However, the nodes are not always anonymous; identity is often known and governed, rendering these restricted node-set networks exposed to a Sybil attack through node collusion or bribery.

### 1.6. Identity & Compliance

If a blockchain network node-set is limited, this implies knowledge of the identity of the node providers in order to govern and enforce the restriction. Some blockchain networks may also require identity for business, compliance and legal requirements (e.g. restrictions on smart contract data storage location and assimilation, commercial sensitivity, data and transaction privacy and counterparty identity).

Anonymous nodes are more secure, but are not available in limited node set environments, and do not fulfil legal and compliance requirements. Conversely, identified nodes are compliant, yet exposed to potential collusion, bribery and fraud.

This is the common paradox that is solved by the Fern infrastructure solution.

### 1.7. Data Oracles

In addition to the aforementioned attack vectors on blockchain networks, we identify a second area where lack of anonymity and / or collusion may lead to fraud: interfaces with external data services that may be intercepted or manipulated by malicious actors.

Blockchain applications typically consume data from a number of external services, including identity services, data feeds, and external payment services. Oracles are implemented where data is consumed by a smart contract from an external service, and proof is required in the blockchain that the response from the service is authentic and untampered.

Two oracle attack vectors are described:

**Interception**. This is where data is requested from a single external data source, but the data received by the smart contract has been intercepted, modified or replaced by a malicious actor.

**Collusion or bribery of data source**. Here the data is authentic and hasn't been tampered with in transit, but the original data source colluded, was bribed or simply acted maliciously to send an incorrect response.

#### 1.7.1. Unsigned Data

Unsigned data from an external source contains no proof of authenticity, and there is an inherent risk of interception.

This interception threat is mitigated by a network of independent oracles that witness and sign the data, acting as a trustless bridge between the data source and the blockchain smart contracts (e.g. ChainLink [12]).

#### 1.7.2. Signed Data

In the longer term, it is highly likely that data sources will provide proof of origin by signing their data with their own private key, enabling a smart contract to independently validate authenticity and tamper resistance, negating the threat of interception.

#### 1.7.3. Oracle Collusion Threat

Oracles must be identified as they are by definition a trusted data source. Nonetheless, there exists a threat of collusion and bribery between the actor standing to gain from a specific response, and the oracle data provider.

An example of this is an independent identity (KYC) check of an individual conducted by a trusted third party. The check is conducted when

a smart contract calls the external identity check service and receives a signed response. If the individual knows the identity of the third party conducting the identity check, they could collude or bribe them to influence a different response (e.g. identity check pass instead of fail).

## 2. FERN DESIGN OVERVIEW

The aim of Fern is to provide a more secure and scalable, yet business-compliant environment for blockchain applications, without compromising the fundamentals of blockchain network security.

We define distributed application networks as limited node set blockchain networks that host one or more smart contract applications. These include private, enterprise, consortium networks, as well as layer 2 networks.

Fern is a community of node, oracle and other services that help decentralise distributed application networks and make them more secure. Fern facilitates the anonymous selection, connection and provision of nodes and oracles by identified and verified community members. Outside networks and applications connect directly to Fern and use its nodes and services.

Fern stores community member identities and enables them to act anonymously in distributed application networks. All member activity, reputation, selection, oracle calls take place on Fern.

Collusion attacks are avoided through random selection of nodes and preserved anonymity of the selected members. Compliance is provided by enabling anyone to pre-validate the public identity of all community members.

Members remain anonymous through the masking of identifiable addresses, while at the same time providing ring signature [13] and other zero knowledge cryptographic proof that all addresses are derived from community members.

Fern's main components include:

1. **Core:** Fern core manages community membership, identity, selection, reputation, oracles.
2. **Passport**. A wallet for Fern community members to manage their identities and provide proof of membership while remaining anonymous when transacting.
3. **Overlay**. A set of components to enable decentralised application networks on any blockchain platform to use Fern.

Fern brings the following benefits to decentralised application networks:

1. **Protection against collusion-based attacks:** Collusion and Sybil attacks on blockchain networks where the number of nodes is limited, and node identities are known (i.e. the majority of deployed commercial blockchain applications).

2. **Decentralized Services Access:** Access to a pool of nodes, oracles and other external services that augment the security and decentralized nature of a network.

3. **Integration**: Infrastructure for standard, secure integration in a decentralised environment.

Fern seeks to address the entire market of commercial decentralised applications, regardless of their underlying technology platform.

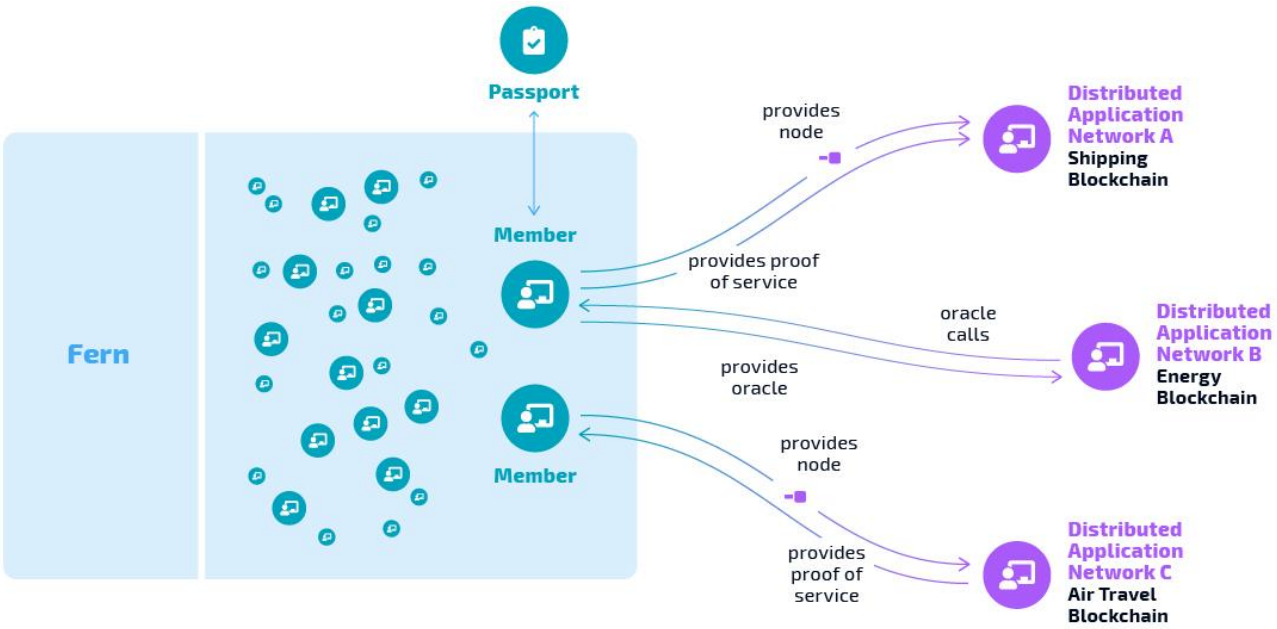



Figure 1: Fern protocol used to decentralise shipping, energy and air travel distributed application networks. Fern protocol randomly and anonymously selects identified, compliant member nodes, oracles and other services.

The following table illustrates the primary categories of decentralised network platforms in the context of decentralised control and potential collusion attack vectors.

| Platform | Category | Public: anyone can send a transaction | Open: anyone can add unlimited nodes | **Anonymous node providers** | **Identity of node providers known** | **Node Collusion Possible** | Networks |
|---|---|---|---|---|---|---|---|
| Ethereum (Bitcoin) | Public | Yes | Yes | **Yes** | **No** | **No** | 1 of each, + forks |
| Polkadot (Cosmos) | Public Interledger | Yes | Yes | **Yes** | **No** | **No** | 1 of each |
| EOS (NEO, Stellar, DASH…) | Semi Public Limited Node Set | Yes | No | **No** | **Yes** | **Yes** | 1 of each. + forks |
| Ethereum (Quorum, Hyperledger Fabric, Corda..) | Private / Permissioned Limited Node Set | No | No | **No** | **Yes** | **Yes** | 1000's |
| **Fern** | - | Yes | Yes | **Yes** | **Yes** | **No** | 1000's |

## 3. Fern Protocol

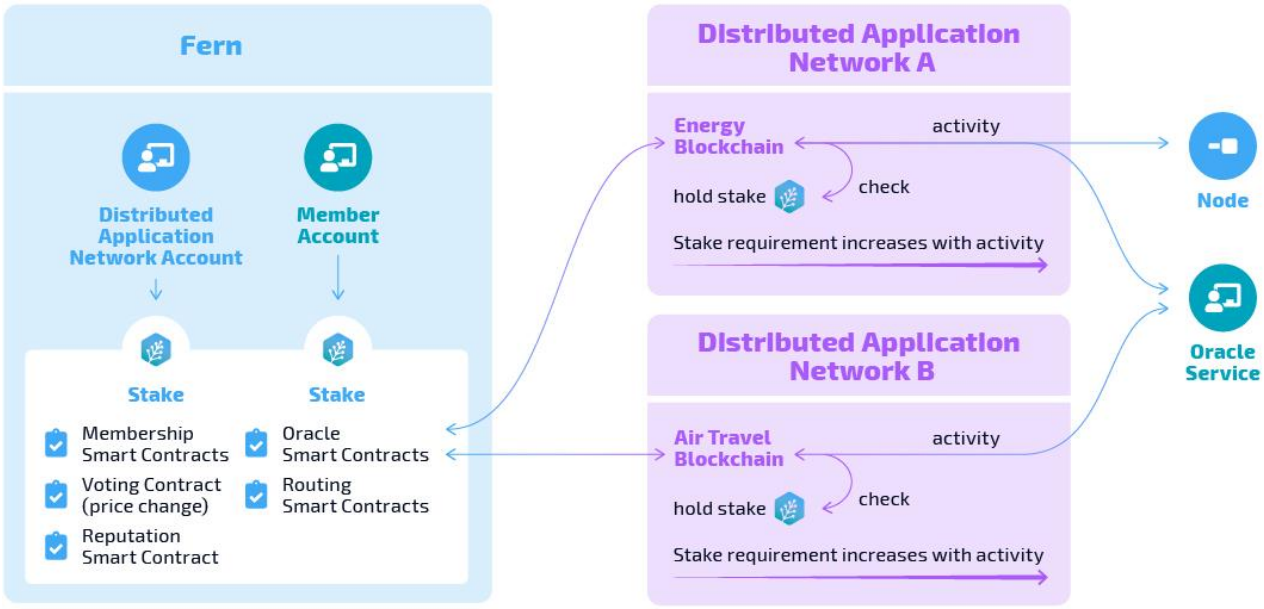



Figure 2: Fern member accounts hold stake. Fern core smart contracts manage membership, governance, reputation, reward, and facilitate inter network and oracle integration. Fern overlay contracts enable distributed application networks to provide proof of activity, manage rewards and contribute to reputation.

### 3.2. Fern Architecture

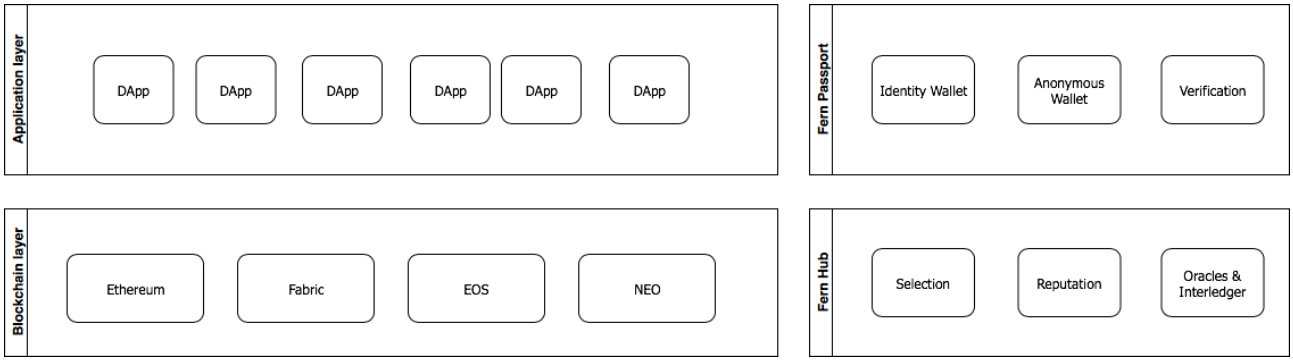



Figure 3: Fern architecture stack defines a standard blockchain stack (left) and integrated Fern components (right) for decentralisation of services. Fern passport provides members with a wallet required to manage public identity and anonymous transactions, as well as verification of other anonymously transacting members. Fern core provides member selection, reputation management, and oracle integrations.

# 4. FERN COMMUNITY

We define the Fern community as a group of independent members providing secure, trusted and compliant blockchain environment components (nodes and oracles) for decentralised application networks.

Fern distinguishes between the concept of a node, and a member. A node is a computer hosting an instance of blockchain software and database, whereas a member is the person or entity that controls the node.

Collusion and bribery are attacks undertaken by malicious *members* providing nodes and oracles, hence the importance of the member identity, as well as anonymity and uniqueness, to prevent collusion.

## 4.1 Identifiable yet Anonymous

Each member has a single, unique Fern community membership.

The membership is recorded in two distinct and unique profiles: *identifiable* and *anonymous*. Each member holds two private keys in their Fern passport wallet, one for their identifiable profile, and the other for their anonymous profile. The two profiles cannot be associated with each other by anyone other than the member themselves.

### 4.1.1. Associating Identity

Member identities in Fern are only validated by the community, and by independent third parties. All new members joining the Fern community must provide and sign a minimum set of identity proofs (e.g. company registration documents, individual government issued identity documents such as passports), as determined by the community from time to time. The identity proofs (signed hashes of the documents) are stored in the Fern core smart contracts and are verified and signed by other community members or third parties, establishing trust in those identities.

All Fern member identifiable profiles are visible to others, and associated identity proofs may be inspected and validated at will. A history of validations is stored in the identifiable profile on the Fern core without revealing the identity of the validators.

#### 4.1.1.1. Anonymous Activity

All Fern activity undertaken by a member, including provision of nodes, oracles, and other decentralised services is conducted using only their anonymous profile. The anonymous profile keeps the usage of nodes and services separate from the identity of the member.

### 4.1.2. Community Membership

All members in the community are part of a cryptographic ring. Members engaging anonymously with each other can prove their membership of the ring, and therefore prove that theirs is one of the identities available for inspection, without revealing their actual identity.

### 4.1.3. Reputation

Each community member has a single identifiable and anonymous membership, and therefore a single reputation across distributed application networks. Reputation is updated based on proof of activity performed by a member anonymously, as measured by the Fern overlay smart contracts deployed in the distributed application network.

The reputation guards against a member acting maliciously within a distributed application network, and then subsequently repeating this behaviour across other distributed application networks.

A reputation score is calculated through the reporting by Fern overlay smart contracts deployed on distributed application networks to the Fern core where member reputation is managed. The reputation is recorded on chain in the Fern core smart contracts.

Node activity (block confirmation and transaction processing) increases positive reputation, whereas missing block confirmation and rejection of signed blocks by the majority penalises reputation. Similarly, successful oracle responses increase positive reputation, whereas lack of service availability and responsiveness penalises oracle reputation.

Random anonymous selection of member nodes and oracles is followed by a reputation filter, such that members below a predetermined reputation threshold will be rejected.

#### 4.1.4. Equilibrium

Distributed application networks seek to increase the number of independent nodes and oracles to raise trust and security for users of their applications.

We expect an equilibrium whereby a distributed application networks select and reward an optimal number of nodes and oracles to demonstrate a relatively trustless network to their users, while sustaining this number through usage of their applications and their underlying application business model.

## 5. PASSPORT

The membership passport is a wallet that holds the private keys for the identifiable and anonymous profiles of a member. It also enables validation of membership, while retaining anonymity when transacting.

An aspect of the solution relates to creating a ring (or group) of prospective node providers $p_1 \dots p_n$ that each reveal their identities $i_1 \dots i_n$ and associated proofs, also referred to herein as their cryptographic passport.

### 5.1 Cryptographic Passport

The passport, a digital wallet belonging to node provider $p_1$, includes a standard private and public key pair $k_{1\ priv}$, $k_{1\ pub}$, a ring member private and public key pair $r_{1\ priv}$, $r_{1\ pub}$ and a ring signature $r_{sig\ 1}$, composed using the ring member private key $r_{1\ priv}$ and all of the ring member public keys $r_{1\ pub} \dots r_{n\ pub}$. The ring signature $r_{sig\ 1}$ should always use all of the public keys $r_{1\ pub} \dots r_{n\ pub}$ in the group in order to maximise anonymity. The key pair $k_{1\ priv}$ and $k_{1\ pub}$ are intended for use in sending and signing blockchain transactions.

### 5.2 Ring Signed Identity Proofs

The ring member private key $r_{1\ priv}$ is used by the node provider $p_1$ to sign its own identity and proofs $i_1$. A feature of the ring signature scheme is that it preserves individual member anonymity such that the ring signature $r_{sig}$ cannot be traced to the ring member private key $r_1\ priv$. The ring signature $r_{sig\ i1}$ created when signing a specific identity proof $i_1$ can, however, be used to verify that a message was signed by a ring member private key.

This feature allows anyone to view the ring member identities $i_1 \dots i_n$, alongside cryptographic proofs $r_{sig\ i1} \dots r_{sig\ in}$ that each is indeed a ring member. The viewer will not, however, be able to link the identities in the ring to their individual corresponding private keys $r_{1\ priv} \dots r_{n\ priv}$ or public keys $r_{1\ pub} \dots r_{n\ pub}$.

### 5.3 Ring Signed Block Signing Proofs

A regular private key $k_{1\ \text{priv}}$ is used by node provider $p_1$ to sign blocks. It may also be used by node provider $p_1$ to sign and send transactions to the blockchain. The key pair $k_{1\ priv}$, $k_{1\ pub}$ remains anonymous to others on the network.

In order for node provider $p_1$ to prove a ring membership it must sign a message (block or transaction) using both $r_{1\ priv}$ and $k_{1\ priv}$.

This is achieved by signing a message $m_1$ (block or transaction) using private key $k_{1\ priv}$, and then signing the signed message $m_{1\ sig}$ using ring private key $r_{1\ priv}$ and all ring public keys $r_{1\ pub} \dots r_{n\ pub}$ to create a ring signature $r_1\, m_{1\ sig}$. This proves that both $r_{1\ priv}$ and $k_{1\ priv}$ are in the hands of node provider $p_1$ and that $p_1$ is therefore both the node signing blocks and/or transactions and a member of the ring, without revealing $p_1$ identity.

### 5.4 Passport Validator

The initiators of a blockchain network, along with their users, may use a passport validator *pv* to validate ring signature $r_x\, m_{x\ sig}$ to verify that the key $k_{x\ priv}$ used by the node provider $p_x$ to sign blocks or transactions, belongs to a node provider that is a member of the ring, such that:

$$pv(r_x\, m_{x\ sig}) = true$$

The same technique is used on chain in a smart contract to prove group membership of the anonymous signatory.

### 5.5. Anonymous Node Selection

An aspect of the solution involves a smart contract algorithm that randomly and anonymously selects a subset of the nodes from the ring (community) to provide nodes to a specific distributed application network. The initiators of the distributed application network will have no influence over selection, nor will they know the identities of the subset of node providers in their network.

This feature provides reassurance to the networks initiators, application developers and users of the blockchain network that the nodes in the network are controlled by providers that are members of the ring (community) where the identity of all members is revealed, while maintaining the anonymity of the node providers in their blockchain network, thus rendering it more

difficult for any party to bribe or collude to conduct a 51% or Sybil attack and defraud and thereby undermine ledger security and integrity.

### 5.6 Anonymous Oracle Provider Selection

An aspect of the solution includes a similar scheme for blockchain oracles, whereby *n* oracles provide an information service to a distributed application network. The oracle providers $o_1 \ldots o_n$ are all members of the ring network and have provided identity proofs signed using their ring signature private key and the public keys of all other ring members.

An aspect of the solution involves a smart contract algorithm that randomly and anonymously selects a subset of the oracle providers from the ring (community) to provide oracle services to a specific blockchain network. Individual oracle queries will be randomly and anonymously routed to one of the selected providers. The initiators of the distributed application network will have no influence over selection, nor will they know the identities of the subset of oracle providers serving their network, nor the oracle provider serving a specific query.

The oracle providers sign their data responses using their ring signature private key $r_{1\ priv}$ and the public keys of all other ring members $r_{1\ pub} \ldots r_{n\ pub}$. The smart contract receiving data from the oracle will validate the ring signature proving that the response came from a trusted oracle that is a member of the ring including identified oracle providers $o_1 \ldots o_n$, without revealing the identity of the oracle.

This feature makes it more difficult for anyone to act maliciously by forcing the trusted oracle $o_x$ to provide a falsified response. This is because without knowing the identity of the trusted oracle $o_x$, it will be difficult to bribe or influence the response, for example bribing an oracle to return a response that a specific's individual's credentials are valid, when they are not.

### 5.7 Avoiding Duplicates Entries

Each member has a pair of private keys, one used to sign its identity and generate the ring signature, the other used anonymously for blockchain transactions. The member uses the former to sign transactions with the latter, in order to prove that the latter, although anonymous, belongs to an identified member of the Fern community.

However, a member could in theory generate two distinct anonymous keys, and generate a ring signature using its identity key for both of them, to prove that they are owned by an identified member of the community. The member reputation is tied to their anonymous key, and allowing multiple anonymous keys enables the discarding of reputation. This is an issue where malicious behaviour has lead to poor reputation, and it in the interest of the community to be aware of this.

This is solved by storing a table of all unique member anonymous accounts in a Fern core smart contract. This table prevents the same party from creating multiple anonymous identities and signing them with the same ring private key. A combination of ring signatures and linked ring signatures and/or zero knowledge proofs are used to ensure the list remains consistent, cannot be manipulated and does not compromise anonymity of the acting members.

| Slot | Anonymous Address | Linked Ring Signature |
|---|---|---|
| 1 | 0x123… | S8D… |
| 2 | 0x456… | 9J7… |
| 3 | 0x533… | 3J3… |

A linked ring signature is stored in this case, such that if the same private key is used to sign two anonymous entries the identity of the ring private key is revealed. This disincentivises registration of multiple anonymous keys by the same member.

It is assumed that the same anonymous address is used by each member across multiple distributed application networks to which it provides service.

## 6. FERN OVERLAY

The overlay provides a set of smart contracts to enable any blockchain technology to integrate with Fern.

The Fern overlay smart contracts are deployed to distributed application networks that are bridged with the Fern core to provide the following distinct functions:

**Meter**. Measurement of services provided: validated blocks for node providers, responses for oracle providers. Meter also measures service performance relative to SLA (e.g. 99% uptime), and influences anonymously stored reputation accordingly.

**Reporter**. Sharing activity proofs with the Fern core in order to update anonymous member reputation.

**Rewarder**. Collection of fees from decentralised application and distribution to nodes, oracles and other service provider anonymous member accounts, ensuring that stake and reputation in Fern are adequate before issuing rewards.

Bridging is achieved using a bridge relay network to transfer signed messages.

## 7. FERN WORKFLOW

This section describes the flow of engagement in the community.

### 7.1 Activities

Fern enables the following activities:

- All members
  - Join
    - Identification
    - Passport creation
    - Reputation initiation
    - Stake deposit
  - Voting
    - Improvement submission
    - Vote
  - View
    - Member identities
- Distributed application
  - Specify requirements
- Provider
  - Specify capability

### 7.2 Identity

Anyone joining the community is required by their counterparts to provide proof of identity. Hashes of the proofs are stored in the blockchain for future validation. Failed identity checks conducted by other Fern community members will be flagged against member reputation (identity validator anonymity will be maintained).

### 7.3 Requirements

A decentralised network application may publish requirements using a smart contract in the Fern core smart contracts:

- Types of service required e.g. node, identity oracle, key recovery
- Max. number of providers required, by type
- Transaction and data projections (optional)
- Unit of reward
- Region (in case restricted)

This will enable distributed applications to decentralise control of their network to anonymous providers without compromising their security nor their regulatory requirements.

For example, network may be restricted to nodes hosted in the EU region for GDPR compliance reasons.

| Service | Node | Identity Oracle | Key Recovery |
|---|---|---|---|
| Max. Providers | 100 | 3 | 3 |
| Max. Response Time | 100ms | 1s | 1s |
| Min. Uptime | 99.9% | 99.9% | 99% |
| Region | Asia | Asia | Asia |

Table 1: Example of distributed application requirements

### 7.4 Selection

A blockchain application specifies its region (if restricted), and providers are randomly and anonymously selected from the identified and verified and community providers.

### 7.5 Manual

We expect some blockchain applications to wish to engage with specific providers for specific services. This will be allowed as long as the number manually selected providers represents a consensus minority for the network, so as to avoid collusion attack vulnerabilities. The ratio of manual vs Fern-selected providers will be available for inspection by users of the decentralised application.

### 7.6 Setup

A limited time window is provided for the node or oracle to be integrated into the blockchain application. This may require additional work on both sides. Failure to integrate post selection within the predefined time window results in a reputation penalty.

### 7.7 Termination

The improper termination of services by a provider must be disincentivised, and will trigger a shut down protocol. A provider terminating their services (e.g. no longer providing their services to a decentralised application) may liquidate their stake, after waiting for a predefined period (e.g.

one month). This is to disincentivise providers from quickly staking and withdrawing - which could have adverse effects on service and provider selection.

Malicious behaviour detected by the protocol (e.g. poor or non-compliant service level by a provider) will result in a reduction in reputation.

# 8. TECHNOLOGIES

## 8.1. Application Agnostic

Infrastructure should be application-agnostic. Decoupling the infrastructure from the application provides mutual benefits for application developers and end users alike.

## 8.2. Blockchain Platform Agnostic

We expect blockchain networks to use a range of blockchain technology platforms. We have already seen numerous competing blockchain network platforms launched and adopted. Fern infrastructure is designed to be agnostic of the underlying blockchain technology choice. This is achieved by providing and supporting a set of overlay smart contracts for a range of ubiquitous blockchain technology platforms and supporting bridging between them.

### 8.2.1. Ethereum & Quorum

In addition to the Ethereum public network, Ethereum software is often used to power private, permissioned, consortium or other limited node-set blockchain networks. Nodes in those networks are identified and the networks are therefore exposed to potential collusion attacks.

Fern may be used to add collusion protection to an Ethereum node set by using the infrastructure to anonymously select from the identifiable community of providers. Similarly, Fern overlay will support oracle requests from Solidity smart contracts.

### 8.2.2. Hyperledger Fabric

The Hyperledger Linux foundation group includes a number of very different blockchain technology platforms including Fabric.

Fabric is designed from the ground up to serve enterprise consortium networks. In the Fabric architecture privacy is maintained between transacting parties using channels. The channels provide mini ledgers between counterparts to a transaction. The shared historical ledger between all parties in a network is known as the ordering service.

In order to increase security through decentralisation in a Fabric network, the channels, and especially the ordering service must be distributed and run with a consensus algorithm that provides Byzantine protection.

If a channel or ordering service is compromised through node collusion, transaction history may be compromised, and fraud may occur (e.g. asset theft).

Fern may be used to add collusion protection to a Fabric ordering service node set by using the infrastructure to anonymously select from the identifiable community of providers. Similarly, Fern overlay will support oracle requests from Fabric chain code.

### 8.2.3. Other Blockchain Platforms

Similarly, Fern open source overlay will be extended to support the most popular blockchain platforms, including Corda, Stellar, EOS, NEO, NEM and many others.

## 8.3. Layer 2

Truly open public blockchains with unlimited nodes struggle to scale, and this is where layer 2 protocols are beginning to be applied. These protocols delegate the high throughput transactions and data to a sub network of nodes.

However, the layer 2 network / side chain is exposed to the same trust and collusion as other limited blockchains. Fern enables a layer 2 network of nodes to be selected anonymously from a known community, thereby reducing the exposure to collusion threats.

### 8.3.1. Plasma

Plasma is a scalability specification similar to side chains involving the locking of assets on a main ledger, and usng this as fallback in case of malicious activity. Plasma builds off chain merkle proofs to provide a more trusted off chain environment.

A plasma environment may be centrally hosted for simplicity and scale. However, this offers no protection against censorship, shutdown, and reorder. If multiple Plasma nodes are required,

then we once again face the threat of collusion. This is where Fern can be used for Plasma node selection, to ensure that a group of nodes add protection, provide compliance if required, yet remain anonymous to protect against collusion, censorship, shutdown and reorder.

#### 8.3.2. Polkadot

Polkadot is an interledger protocol and bridge network initially for Ethereum compatible blockchains.

The blockchain networks in Polkadot are known as parachains. The parachains' headers are sealed within a relay-chain block ensuring no reorganisation, or "double-spending", is possible following confirmation. Each Parachain will need to be defined in the Polkadot Parachain Registry.

Fern may be used for selection of Parachain node providers to deliver additional security through reduced likelihood of node collusion, as well as compliance, that may be required where a Parachain processes sensitive data, through identified yet anonymous node provider engagement.

#### 8.3.3. Bridging

An inter ledger bridge is required between the decentralised application networks and the network hosting the Fern core smart contracts.

#### 8.3.4. No Bridging

Some blockchain networks, due to the security and/or regulatory environments in which they operate, will not be able to practically and physically connect to another blockchain.

In this case where bridging using an interledger protocol is not possible, proofs of service may be derived from the blockchain networks, signed by the nodes, and presented independently to the other network smart contracts. These will verify the proofs and signatures and update the stake and reputation accordingly.

### 8.4. Oracles

Fern will provide efficient, more secure access to off-chain oracles. Each oracle type (e.g. KYC, AML, pricing, weather) will have a generic interface defined in Fern overlay smart contracts. Decentralised applications call these overlay interfaces directly from their own smart contracts.

The Fern overlay contracts call the Fern core via an interledger bridge, where a gateway contract selects and calls out to a specific KYC API via a Fern oracle service.

The Fern oracle services, are stateless, keyless, off-chain oracle adapters that provide an interface between Fern gateway smart contracts and the web interface of each external service provider.

The web interface will provide an anonymous ring signed response, independently verifiable on chain by a smart contract.

## 9. SERVICES

Fern may also be used to increase security by reducing the threat of collusion attacks for range of other services:

### 9.1. Revealers

Some restricted node set blockchain networks do not provide the same level of visibility and transparency as open public networks, where anyone can download, host, validate and interrogate the ledger. In order to compensate for this lack of visibility and transparency, we introduce an independent set of transparency and visibility providers referred to as revealers.

In order to add security, and be independent of the network, the revealers must come from an identified trusted pool, while remaining anonymous. Fern infrastructure provides the ability to select random revealers for a network.

### 9.2. Identifiers

Numerous blockchain applications require identity verification of some form, and therefore consume a range of identity oracle services. The Fern infrastructure includes a set of standards for recording and integrating identity proofs and attestations.

Verifiers may collude or be bribed by those they are verifying. Fern infrastructure provides a method for selecting independent verifiers.

### 9.3. Retrievers

Blockchain applications delegate security to their users by providing them with their own keys. These users may require a key recovery service in case they no longer have access to their key wallet. Recovery of such keys is a problematic area in terms of user experience and trust, and we envisage a number of key recovery providers independently, or as a group, offering advanced decentralised key recovery features to blockchain applications and their users. Those key recovery services would be more secure, and less vulnerable to collusion attacks if selected anonymously using Fern infrastructure.

### 9.4. Managers

Some blockchain applications run on isolated networks. Creating and managing an isolated network, including management of a VPN and node / IP whitelist should ideally be performed by a series of independent parties. This will help to preserve anonymity of selected providers. Manager need to be randomly selected. To further secure the network, multiple network providers should be employed, where each only has knowledge of immediate neighbours, and IP addresses are obfuscated.

### 9.5. Guardians

Numerous solutions are proposed and available for solving data privacy in blockchain smart contracts. Some, such as sMPC (Simple Multi Party Computation) schemes (e.g. Buterin 2014 [1], Enigma [2], Hawk [3] 2016, Keep [4] 2017, VIFF [5]) require independent third parties to host and process subsets of data in off-chain networks. Where this service is required, providers may propose their privacy protocol to blockchain networks and applications. If these provides are members of the Fern community, they remain anonymous, yet identifiable for compliance, reducing likelihood of collusion and fraud in those systems.

### 9.6. Hosts

Server based applications for distributed systems underpinned by a blockchain network should be hosted by multiple independent parties to provide high availability and avoid a single point of failure or attack (e.g. ability to shut down service). These are generally keyless, and stateless to reduce their control and attack vectors. Fern infrastructure may be used to select server providers, and thereby reduce the ability of such parties to collude to shut down or otherwise act maliciously.

### 9.7. Processors

In order to counteract the computational bottleneck that many blockchain smart contract based applications deal with, blockchain applications have the option to utilise decentralised processing services (e.g. Truebit), enabling off chain trustless smart contracts to securely perform computational tasks. Fern

infrastructure may be used to reduce the ability to collude among processors.

### 9.8. Containers

Blockchain application data storage services (e.g. IPFS) may be offered by providers of storage nodes. Dispersing data across a number of providers increases resilience and trust. Collusion threatens to undermine the security of distributed storage, as nodes may collude to delete data, for example. Fern infrastructure may be used to select the storage node providers anonymously, yet from an identified and trusted group to ensure compliance.

## 10. REFERENCES

[1] Gavin Wood. Polkadot: Vision For a Heterogeneous Multi-Chain Framework (white paper). 2017.
[2] Polkadot – What is it? https://polkadot.network/#cover
[3] Vitalik Buterin. DevCon3. Nov. 2017.
[4] Jae Kwon, Ethan Buchman. Cosmos – A Network of Distributed Ledgers (white paper). 2017.
[5] Proof of Authority Chains – Wiki, Parity Ethereum Documentation. https://wiki.parity.io/Proof-of-Authority-Chains
[6] IPFS. https://ipfs.io/
[7] Filecoin. https://filecoin.io/
[8] Community Token Economy Whitepaper 1.0.1
September 2018 (p9).
[9] Above [8] (p10).
[10] Prisoner's dilemma. https://en.wikipedia.org/wiki/Prisoner%27s_dilemma
[11] Plasma. https://plasma.io/plasma.pdf
[12] ChainLink, A Decentralised Oracle Network (white paper) - Steve Ellis, Ari Juels, and Sergey Nasarov, 4 September 2017 (v1.0)
[13] https://en.wikipedia.org/wiki/Ring_signature, https://github.com/vwvw/Ethereum-RingCT
[14] One-time Ring Signatures, Cryptonote White Paper
[15] AOS Ring Signatures
[16] Linkable Ring Signatures
[17] An Investor's take on Cryptoassets, John Pfeffer. https://s3.eu-west-2.amazonaws.com/john-pfeffer/An+Investor%27s+Take+on+Cryptoassets+v6.pdf
[18] How to Leak a Secret, Ronald L. Rivest, Adi Shamir, and Yael Tauman https://link.springer.com/content/pdf/10.1007%2F3-540-45682-1_32.pdf
[19] The Meaning of Decentralization, Vitalik Buterin. https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274
[20] EOS, https://eos.io/
[21] Ethereum Classic 51% attack https://cointelegraph.com/news/ethereum-classic-51-attack-would-cost-just-55-mln-result-in-1-bln-profit-research