Comprehensive analysis of frontier research related to Self-Sovereign Identities

Matthew David Lowdermilk
(唐无名)

June 2021

Comprehensive analysis of frontier research related to Self-Sovereign Identities

Candidate..	Matthew David Lowdermilk 唐无名
School of Study..............................	School of Computer Science
Student ID Number.........................	3820191014
Supervisor......................................	王树良
Chair of Defense Committee..........	
Degree Applied..............................	Master of Computer Science
Major..	Computer Science and Technology
Date of Defense...............................	June 2021

Declaration of Originality

Authorization Statement

Acknowledgments

I have been brought to this point due to the support, influence, and contribution of many people, contributing in their way. In particular, I would like to thank: (1) My family that supported my education and I accompanied with on our first journey to China in 2000. (2) Lin Crowley and her encouragement in improving my Chinese during my undergraduate education. (3) The Confucious Institute that sponsored my first visit to BIT as a month-long Chinese language program during summer 2018 ~; without that visit, I would not have known of BIT or been kept informed for future opportunities here. (4) My professors and staff at BIT for their advice, support, and inspiration adapting to my studies. (5) The People's Republic of China and the China Scholarship Council for their gracious acceptance of me for this scholarship program and accomodating and welcoming nature.

I am grateful for all the intelligent and kind friends I have met during my studies at BIT. And the advice from second-year students - particularly Ian Wesson (安龙) and his advice to use HTML/CSS to layout this thesis - and so many more whose words and actions brought me to this point.

I am sincerely grateful to all who aided me in getting here.

- Matthew David Lowdermilk (唐无名)

Abstract

Distributed Ledger Technology (DLT) refers to the broader class of distributed ledgers that include blockchain and directed acyclic graphs. Blockchain, the most familiar form of DLT, has transitioned from cryptocurrencies to Smart Contracts and current further development resolving deficiencies in scalability, interoperability, sustainability, privacy, and governance.

The Distributed Identity Foundation, with members including Microsoft and IBM, is working with the W3C and the Linux Foundation's Hyperledger project to develop a unified, interoperable ecosystem allowing users to control and store identity information by creating a decentralized digital identity or Self-Sovereign Identity (SSI).

Using DLT, a much-needed evolution of the internet will create a common identity layer allowing people, organizations, and things to have their own discrete SSI. This will alter the future of big data, giving control of identity back to the owners, allowing full regulation of who can use their data and how it can be used.

The aim is to create a new system of applications and products that enable developers and businesses to offer solely client-side services—giving every person their own portable digital identity securely stored to preserve privacy.

This research contributes a Comprehensive analysis of Distributed Ledger Technology, specifically the generations of blockchain and limitations that are being resolved. The analysis then covers the parts of an SSI: Decentralized Identifiers, Verifiable Credentials, and Identity Agents, and documents current development by Hyperledger Identity Stack, Atala Prism, and the Decentralized Identity Foundation.

Key Words: Distributed Ledger Technology; Blockchain; Blockchain 3.0 – Scalability, Interoperability, Sustainability, Privacy, and Governance; Self-Sovereign Identities; Decentralized Identifiers; Verifiable Credentials; Identity Agents; Hyperledger Identity Stack; Atala Prism; Decentralized Identity Foundation;

Table of Contents

1 - Introduction

We live at a time where the world is driven by big data, transforming systems across society, including traffic, health, government, logistics, and national defense. This data is currently concentrated in “islands” by governments, internet giants, and communications institutes. However, these islands lack data credibility as there is no quality verification, and as such, they are not conducive to scientific research [Li_2017].

Distributed Ledger Technology (DLT) drew attention more recently, providing tamper-proof distributed ledgers verified by a trustless consensus of the system’s participants. DLT technology is a global foundation for building cryptographic digital trust between companies, individuals, and machines - Trust over IP [ToIP_2020], removing the necessity for trusted third-party intermediaries.

Blockchain (BC), the most familiar form of DLT, has transitioned from cryptocurrencies (Blockchain 1.0) to Smart Contracts (Blockchain 2.0) and further development resolving deficiencies in scalability, interoperability, sustainability, privacy, and governance (Blockchain 3.0). One major change that Blockchain 3.0 aims to bring will alter the future of big data, giving control of identity back to the owners, allowing full regulation of who can use their data and how that data can be used. This will require a change or replacement of existing big data systems [Karafiloski_2017].

The Distributed Identity Foundation, with members including Microsoft and IBM, is working with the W3C and the Linux Foundation’s Hyperledger project to develop a unified, interoperable ecosystem allowing users to control and store identity information by creating a decentralized digital identity or Self-Sovereign Identity (SSI). The aim is to create a new system of applications and products that enable developers and businesses to offer solely client-side services. "Every person has a right to an identity that they own and control, one that securely stores elements of their digital identity and preserves privacy [Microsoft_2019]."

The rest of this paper is organized as follows. Section II covers the fundamentals of DLT and subtypes BC and DAG. Section III explores BC generations, cryptocurrency (v1.0), smart contracts (v2.0), and work in progress on scalability, interoperability, sustainability, privacy, and governance (v3.0). Section IV describes the components of Self-Sovereign Identities: Decentralized Identifiers, Verifiable Credentials, and Identity Agents. SSI under active development is reviewed in Section V, followed by the conclusion.

Chapter 1 Takeaways

- Big data is currently concentrated in pockets of information controlled by different entities. Such data lacks credibility and is not conducive to scientific research.
- DLT provides a distributed ledger with a trustless consensus mechanism between the system’s participants, allowing a global foundation for building trust between companies and individuals – Trust over IP.
- The Distributed Identity Foundation, W3C, and the Linux Foundation’s Hyperledger project are working on a unified global distributed digital or Self-Sovereign Identity system, aiming to change the client-server web that we know to one that offers solely client-side services.

2 - Distributed Ledger Technology Fundamentals

DLT has come to describe a broader class of technology that includes BC and directed acyclic graph (DAG). The most popular definition of a BC is "a distributed or decentralized ledger" [Frizzo-Barker_2019]. However, BC is a subset of distributed ledgers.

Distributed ledgers offer a decentralized infrastructure realizing 'a single version of truth' in a linked record from the chain's origin - referred to as the genesis block. This is done without the need for a central authority as all nodes in the peer-to-peer (P2P) network share the same full copy of the database (ledger), thereby avoiding a single point of failure (SPF) [Zachariadis_2019].

Distributed ledgers come with permissionless and permissioned transaction validation (permission to write) and public and private access (permission to read), as seen in Table 1. A company internally may choose to use a permissioned ledger with either public or private visibility, while most cryptocurrencies operate as public permissionless ledgers. Consortium ledgers are another type of permissioned ledger with equally qualified nodes often held by different entities (i.e., corporations).

		Write	
		Permissionless	**Permissioned**
Read	**Public**	Bitcoin Ethereum IOTA Cardano	Hyperledger Indy
	Private	N/A	Hyperledger Fabric Corda

Table 1: Types of DLT

2.1 - Blockchain

BC is the most well-known form of DLT. Fundamentally, a BC offers a tamper-proof distributed ledger, or database of records, where each transaction stored is verified by consensus of the nodes (participants in the system). Transaction data is intended to be immutable once added to the chain. One party to a transaction initiates a process by requesting validating nodes to include that transaction in a block and broadcasting the block to the network. Once a block is verified through consensus of the nodes, it is added to the chain and stored across the web, resulting in a unique record with a unique history.

On a BC, a data block is divided into two parts: header and body. The body's transactions are each hashed, and the Merkle Root [Merkle_1990] is the combination of all sub hashes. The header contains the hash of the previous block as well as the Merkle root of the body. This allows for the chaining of blocks and confirmation that all sub-data is verifiably unchanged, see Figure 1.

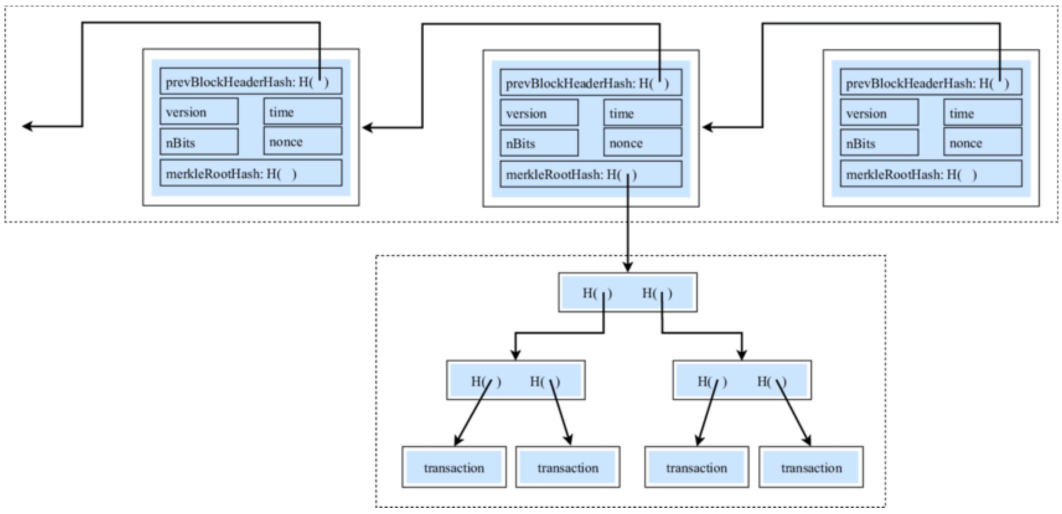


Figure 1: Simplified blockchain block structure [Frauenthaler_2019]

2.1.1 - Means of Consensus

Public BCs are distributed and not controlled by any single party. Ways to ensure that nodes are not malicious, contain accurate information, and contain a single accepted history attempt to resolve the Byzantine Generals Problem [Lamport_2019]. Bitcoin's innovative attempt offered a means of consensus (MoC) called Proof of Work (PoW). While PoW is useful when a BC has a limited network, and many cryptocurrencies have used PoW in their nascent stage (e.g., Ethereum), other attempts to resolve drawbacks of PoW are being tested by other BCs [Lee_2019]. These include Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Delayed PoW (DPoW), Proof of Burn (PoB), Proof of Capacity (PoC), Proof of Communication (PoC) [Chen_2019], and Proof of Activity (PoA) [Liu_2019]. Varieties specific to permissioned chains: Proof of Elapsed Time (PoET), Tendermint, Practical Byzantine Fault Tolerance (PBFT) [Viriyasitavat_2019], and others [Zheng_2017].

Proof of Work

PoW requires miners to attempt guessing a nonce (value) that, when combined with the SHA-256 hash of the proposed block, results in a leading number of zeros. The first miner to correctly compute the nonce is given a predetermined prize of coins. The number of zeros required is updated according to the miners' (nodes') average computing power, allowing the approximate creation of one new block every 10 minutes. Due to the previous block's hash value being included in each new block's header, the new nonce values for any change would have to be updated in the entire chain onward from the altered block. The computing power required for this is realistically unfeasible unless more than 50% of nodes collaborate maliciously.

However, Bitcoin currently allows application-specific integrated circuits (ASICs) to be used in mining. This enables mining pools to work collaboratively and thus gain a large majority of hashing power, potentially altering the chain in their favor. China hosts the largest Bitcoin mining pools, over 79% total hash power, yet no individual pool is over 25%.

Since PoW resembles a lottery for the winning miner if more than one miner guesses the correct nonce a fork could be created. Bitcoin deals with this issue by having honest miners choose to append new blocks and validate the longest chain. Alternate correct guesses that have a shorter chain, and are unused, are called uncles.

PoW has the current drawbacks:

- Waste of electricity, as miners spend large amounts of computing power for essentially a lottery ticket [Stoll_2019, Sutherland_2019]. However, as pointed out in [Frizzo-Barker_2019], creating fiat currency also uses significant power.
- Slower throughput. Bitcoin has the difficulty set to one block every 10 minutes corresponding with current computing power.

Proof of Stake

PoS is the next best-known and used MoC. In PoS, the amount of stake (coins) that a stakeholder has is used with a randomized process to elect which stakeholder produces a block. Successful block creation allows the block producer to gain interest on the amount staked. More public BCs are moving toward PoS as it wastes less electricity than PoW. PoS is being integrated into Ethereum 2.0 in Serenity, its first stage also known as "beacon chain." However, the Ethereum Virtual Machine (EVM), the virtual machine running smart contracts, still operates on Ethereum 1.0, which uses PoW.

Some PoS systems solve honest behavior by locking the stake, requiring stakeholder honesty in block creation, or their stake will be destroyed/slashed (Ethereum 2.0). In the case of stake locking, some chains require a specific term of time that the coins are frozen, decreasing the cryptocurrency's liquidity. Other chains require a specific quantity of coins to participate in staking (e.g., Ethereum 2.0 requires 32 ETH to participate in staking), thus excluding those with a small amount of coins.

The Cardano BC uses Ouroboros, a version of PoS for consensus [Kiayias_2016]. In Ouroboros, a randomized process is used to elect a stakeholder to produce a block based on the stake's weight. Since individuals typically do not have a significant enough stake to be elected, they can delegate their stake to a stake pool, thereby collectively participating in a large enough stake for the stake pool to be elected. There is no minimum stake through this system, no locking of coins, and no slashing required. ADA, the name for coins on the Cardano BC, is freely spendable (i.e., liquid) at any time and can be redelegated from the end of the current epoch plus one epoch (~6-10 days).

Delegated Proof of Stake

DPoS is similar to PoS with 'delegated' meaning that a smaller group of elected members performs validation rather than the entire group. There is a random election of delegates favoring those with a larger amount staked. Block producers are responsible for grouping transactions and broadcasting them to the network, receiving rewards for progressing the network.

Practical Byzantine Fault Tolerance

PBFT is unique in that only one network member becomes the primary participant who communicates with all other participants to validate a specific block, removing the chance of a fork. PBFT and its variants are widely used in consortium (private) chains with foreknowledge of a fixed (smaller) number of known participants. They can tolerate up to a third of participants having any form of Byzantine fault [Castro_1999].

2.2 - Directed Acyclic Graph

Another form of DLT, DAG is a topological directed graph that prevents closed loops. Topological ordering requires that the starting vertex comes before the ending vertex, thus eliminating loops. Rather than building a chain of blocks linked to the previous one, a new transaction must link to one or two other transactions to be processed and confirmed. Only newly added transactions are confirmed, indirectly confirming the entire transaction history.

Notably, with a DAG, there is no global network state. Each network participant stores only their neighbors' local data, relying on other regions to do the same. This has advantages as there are low resource requirements. Transactions can be added in parallel, thereby increasing the overall speed.

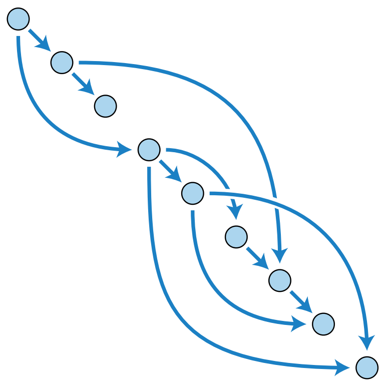


Figure 2: Topological Ordering DAG [David Eppstein]

Chapter 2 Key Takeaways

- Distributed Ledger Technology (DLT) is the broader class of distributed ledgers that include blockchain and directed acyclic graphs (DAG).
- DLT realizes 'a single version of truth' linked from the origin (genesis block) in a P2P network with no central authority.
- Distributed ledgers come with permissionless and permissioned validation (write) and public and private access (read).
- Blockchain is an append-only chain of transactions with each block header containing the previous block's hash.
- Blockchain block headers contain the Merkle hash of all transactions in the block body, providing validity to unchanged transactions.
- Blockchains are distributed and not controlled by any single party. Common means of consensus that ensure that nodes contain accurate information include PoW, PoS, DPoS, and BFT.
- A Directed Acyclic Graph (DAG) is a topological directed graph, ordered temporally forward, thereby eliminating loops.

3 - Generations of Blockchain

BC takes its place as the fifth disruptive computing paradigm following mainframes, PCs, the Internet, and mobile/social networking [Swan_2015]. BC's generations started with Bitcoin, a cryptocurrency enabling transactions between two parties without a trusted third party: Blockchain 1.0. Adding terms and conditions to cryptocurrency transactions by embedding a programming language, Ethereum took its place issuing in Blockchain 2.0. Blockchain 3.0 is a catchall term relating to work in progress to ensure sufficient infrastructure for mainstream use of BC, solving scalability, interoperability, sustainability, privacy, and governability.

3.1 - Blockchain 1.0: Cryptocurrency

Bitcoin, the seminal BC, was designed to be a decentralized P2P electronic cash system, enabling pseudonymous transactions through advanced cryptography without requiring third-party authorization [Nakamoto_2008]. This type of decentralized electronic cash system is termed a cryptocurrency.

A BC network relies on asymmetric encryption - generating two ciphers, or key pair: public key to encrypt and private key to decrypt. Information encrypted with the public key, which can be shared with others, can only be unlocked with the private key, which should be kept private. The private key cannot be calculated via the public key; hence it is safe to release the public key to the public [Guo_2019]. The pseudonymity of transactions occurs as transactions are listed as between the sender and receiver's public keys.

While Bitcoin offers pseudonymity, users are not entirely anonymous. If a user posts their public address (key) on social media, their transactions can be monitored even with the use of Coinjoin or another mixer. If a user desires to use cryptocurrency for the sake of anonymity, such as through utilizing Tor, Bitcoin should be used with caution. Suppose a user chooses to use Bitcoin for the sake of anonymity. In that case, they should not post any knowledge of their account on any social media as it can be traced in logs that are publicly accessible. Transactions on BCs (www.blockchain.com/api) include both the timestamp of the transaction as well as the relay IP address. [Cui_2019, Jawaheri_2020].

3.2 - Blockchain 2.0: Smart Contracts

SCs were introduced by Nick Szabo in 1996 [Szabo_1996] as a means to facilitate, verify, or enforce the terms of an agreement without the need of a third party, giving an example of a vending machine [Szabo_1997]. Being the forerunner of BC, Bitcoin implemented simple SCs on a stack-based design. However, Blockchain 2.0 typically refers to Ethereum and the innovation of Turing-complete SCs on the Ethereum virtual machine (EVM) [Wood_2014]. EVM set the stage with smart contracts growing more complex and intricate. Dapps, DAOs, DACs, and DASs are some examples, with BC transactions no longer restricted to payments and currency transfers [Swan_2015].

- Decentralized application (Dapp): A Dapp is an application that runs on a network in a distributed fashion with participant's information securely protected. Operation execution happens automatically at the fulfillment of preset conditions.
- Decentralized autonomous organization (DAO) and decentralized autonomous corporation (DAC): DAOs and DACs are more complex forms of decentralized apps. To become an organization or corporation, a Dapp might adopt a constitution that would outline its governance publicly on the BC, defining a means to finance its operations.

- Decentralized autonomous societies (DAS): DAS refers to the idea of self-bootstrapping software to crowdfund itself based on a mission statement.

Ethereum is the most well-known BC that works with SCs, and it is currently #2 in market capitalization. However, it has some security bugs:

- **Time Order Dependent (TOD) contracts:** If two submissions to the same contract are order dependent but come at a similar time, they may be included by the miners in a different order, such affecting the resulting payments. (e.g., X is a computation by a user for a prelisted prize, and Y is an update by the designer of the contract for the resulting prize. If both X and Y are submitted around the same time, and say Y is updating the prize to zero, if they are submitted in reverse order, the user who submitted X would lose their prize.)
- **Timestamp dependent:** Contracts may use the timestamp as a trigger condition to execute some critical operations. However, the miner sets the timestamp for the block, ordinarily using the miner's system's current time. The miner is permitted to vary this value by roughly 900 seconds, allowing, in an example of a jackpot contract, the miner to bias the results in their favor.
- **Mishandled exceptions:** In Ethereum, contracts are permitted to call other contracts. If there were an exception (e.g., not enough gas or exceeding call stack limit) in the called contract, it would terminate and return false. The calling contract may not receive the propagated exception depending on whether the caller explicitly checks. This inconsistency can lead to incorrectly handled exceptions.
- **Reentrancy vulnerability:** In Ethereum, when a contract calls another, the calling contract waits for the call to finish. This can lead to an exploit when the caller makes use of the intermediate state.

Other concerns with SCs can be found in [Giancaspro_2017].

SCs in Ethereum are built using a custom Javascript-like object-oriented, state-based programming language, Solidity. There is current work on Vyper, a non-Turing-complete Python-like deterministic language that will come into play in Ethereum 2.0. Ethereum 1.0 uses PoW, but it is transitioning to PoS in its 2.0 version. Ethereum 2.0 will come in three stages, Phase 0, 1, and 2. Phase 0, the Beacon Chain, storing and managing the registry of validators and deploying the PoS consensus mechanism, went live in December 2020. However, SCs are still run on the main chain (the original PoW chain) till Phase 1, due in 2021, comes out.

Cardano SCs are scheduled to go mainnet and be available at the end of July 2021, following the testnet release and active development at the end of April to early May 2021. The SC programming platform for Cardano is called Plutus [Input-Output-HK], based on Lambda Calculus and Haskell. Haskell is a purely functional programming language, offering deterministic stateless functions typically with no side effects. All code is written in Haskell, with off-chain code compiled by Haskell's GHC compiler and on-chain code with the Plutus compiler. In addition to the Plutus platform, Cardano offers Marlowe, a special-purpose programming language for financial contracts. The Marlowe playground is available for writing Marlowe contracts in several ways, allowing the analysis of smart contract behavior by simulating participants' activity. Marlowe contracts can be written directly as Marlowe text or with Blockly, a visual drag and drop programming tool for non-developers. Alternatively, as Marlowe is embedded in Haskell and JavaScript, those programming languages can be used initially and converted to Marlowe and Blockly through the Marlowe playground [iohkdev.io].

3.3 - Blockchain 3.0: Scalability, Interoperability, Sustainability, Privacy, and Governance

There are many limitations with BC 1.0 and BC 2.0 prerequisite to bringing BC into widespread use. Most notably, scalability, interoperability, sustainability, privacy, and governance, which BC 3.0 aims to resolve. There are many competitors to that title, but as of yet, solutions are only theoretical as they are under construction. Some of the techniques designed to resolve these limitations, and BCs that are developing them, will be listed here.

Scalability

The most significant hindrance of BC's largescale use is transaction speed: Bitcoin processes 7 transactions per second (tps), and Ethereum can process 20 tps. At the same time, PayPal can handle almost 200 tps, and VISAs claims to handle over 20,000.

One way to increase transaction speed is through the use of sharding. Sharding has been used by giant databases worldwide since the late 1990s. It is a database partitioning technique that spreads the load over a decentralized network by portioning data through shards or horizontal portions of data. Essentially transactions would occur simultaneously and parallel on each shard could, horizontally scaling the tps. Blockchains using PoW cannot implement sharding. In the potential case of a PoW BC with 100 shards, a bad actor obtaining 1% hash power of the whole network could control 100% of a single shard. However, in a PoS BC, a malicious user must obtain over 50% of the cryptocurrency to initiate an attack. Thus, PoS chains are required for the use of sharding.

One of the first BC platforms to implement sharding was Zilliqa [Zilliqa_2017]. During the testnet stage, the Zilliqa network could process over 2,800 tps. Phase 1 of Ethereum 2.0, scheduled for sometime in 2021, will see the blockchain split up across a proposed 64 shards, increasing the throughput 100-fold. The Cardano BC announced its scalability solution, Hydra protocol [Chakravarty_2020], as part of their fourth era (Basho era). Each head of the Hydra protocol (each shard) can handle around 1,000 tps, resulting in the Cardano BC potentially reaching 1 million tps. The IOHK team, Cardano developers, introduced extended UTXO (EUTXO), an extension of Bitcoin's UTXO model, allowing sharding of stake space without the need to shard the ledger itself.

Additionally, several layer two fixes are being developed to integrate with existing blockchain protocols, allowing off-chain transactions on a P2P network to reduce on-chain bloat and increase throughput.

Lightning Network is a layer two solution built for Bitcoin [Poon_2015]. However, Bitcoin only permits basic SCs, and while it is more scalable using Lightning Network, it is not a contender for the BC 3.0 title.

Ethereum 2.0 has multiple types of layer two solutions, including Rollups (zk-Rollups and Optimistic rollups), Plasma, and others [Wackerow_2021]. Rollups allow transaction execution outside layer one, data or proof of transactions on layer one, and a rollup smart contract on layer one that enforces correct transaction execution through using the transaction data on layer one. Operators are required to stake a bond in the contract. Zk-rollups run computation off-chain and submit a validity proof to the chain. Optimistic rollups assume valid transactions and compute a fraud proof if challenged. Plasma allows layer two blocks on top of the Ethereum BC in the form of multiple side chains. However, these plasma chains only support basic token transfers, swaps, and a few other predicate logic-based transaction types. [Jones_2019].

Hydra is Cardano's layer two solution, allowing coins from multiple parties to be transferred to the second layer Hydra head for rapid processing. Once the head closes, the layer one redistributes the resulting location of the same quantity that was first transferred to the head when it opened. Hydra uses state channels that allow a smaller group of parties to agree on without interaction with the underlying BC. These state channels also enable the execution of SCs using as-is the SCs written on layer one.

Another way to increase the throughput is to use a DAG in place of a BC. IOTA is designed for the Internet of Things (IoT), offering feeless transactions. IOTA's transaction settlement and data integrity layer is called The Tangle. Instead of miners validating transactions, network participants are jointly responsible for transaction validation. Each new transaction must confirm two already submitted transactions to be validated. Every transaction requires computational resources based on PoW algorithms to find the answer to a simple cryptographic puzzle. IOTA currently requires a centralized coordinator node operated by the IOTA foundation. Since IOTA is still a small-scale project, it is uncertain whether it will be successful in practice.

Interoperability

To gain mainstream adoption, existing cryptocurrencies must work together. There are over four thousand blockchains, and communication between them is necessary. Interoperability can be attributed to the likely case that there will not be solely one BC in the end. Numerous contenders, including Wanchain, Polkadot, and Cardano, are attempting different avenues for interoperability.

The goal is to present a mechanism to transfer data and assets between blockchains and legacy systems in a decentralized manner. As it is difficult for BCs to understand one another, significantly more so between any BC and legacy systems like banks, currently, exchanges are a centralized means of communicating between fragmented stores of value. These exchanges are fragile to being hacked or shut down to regulation incompatibility, and as such, are not a stable resource for a decentralized ecosystem. Thus, third-generation BCs must supply a means for reliable cross-chain transfers without needing a trusted third party.

Atomic swaps use a SC to exchange one cryptocurrency for another without centralized intermediaries (e.g., exchanges). This provides one way to exchange cryptocurrencies between the fragmented ecosystem of cryptocurrencies. However, this does not permit exchanges to legacy systems.

Wrapped coins are a form of ERC-20 token, or fungible token, representing an asset (e.g., WBTC) with a similar price to that of the original asset (e.g., BTC) on another smart contract capable chain, so far Ethereum. Typically, a wrapped coin is backed one-to-one with the underlying asset or through a SC. Stable coins are one of the first types of wrapped assets, being fiat-backed (e.g., Tether (USDT)), which has its price pegged to the US dollar). Wanchain uses a similar procedure by locking tokens on the original chain and using proxy tokens for blockchain interoperability.

Polkadot created a network protocol to transfer all data across public permissionless as well as private permissioned BCs. The relay chain, the heart of Polkadot, connects to Parachains, sovereign BCs, and Parathreads, economical BCs that don't need continuous connectivity, using Bridges to connect and communicate. Polkadot does not have a SC platform itself. Instead, it links to Parachains that do, allowing full interoperability and data transfer between chains [Polkadot_wiki].

Interoperability between BCs is one matter; adding additional programming languages to the writing of SCs is another. The KEVM testnet was released in 2018, enabling a correct-by-construction version of the EVM specified in the K-framework. This allows SCs written for EVM to be rigorously proved that they work correctly and can be run on the Cardano BC. The Cardano BC SC layer is scheduled to be released in iterative stages from the end of April through September 2021. The testnet for Plutus SCs is expected at the end of April to early May, followed by Q&A testing and creation of NFT marketplaces, etc. Around the end of June it will be feature-frozen for four weeks for partners (Coinbase, Binance, Yoroi, etc.) to integrate infrastructure before the hard-fork to mainnet.

In September, the first version of the translator from LLVM to IELE without library support is scheduled. IELE is a K-framework mapping supporting a register-based virtual machine. The K-framework is a more mathematical higher-level mapping between the programming language and the LLVM lower-level / machine language mapping. The LLVM backend for K means that any programming language defined in K can be automatically translated to the LLVM. As IELE was defined in K, therefore, it has an automatic translation to LLVM. Essentially, through IELE, using any programming language directly, developers can write, compile and execute SCs [IOHK_March2021].

Sustainability

BCs are not static and require further development to prevent becoming obsolete. A means to fund future developments is necessary for a worthy BC. ICOs are frequently used but are a finite means to generate a large flood of money. They depend on proper management by the team of developers, and there may be problems when the fund runs out. Patronage is another way that arguably leads to centralization, leading to an imbalance in the amount of influence over the system’s evolution and growth.

Additionally, a treasury system can be used, allowing for a percentage of rewards to be sent to a decentralized account that can then be democratically distributed by funding proposals voted on by the token holders. The treasury model is robust. It provides an avenue to be refilled proportional to the currency’s size while providing democratic participation for stakeholders to choose what they think holds higher priority to strengthen the system. The challenges include:

1. Ensuring that the voting system is fair and proper.
2. Providing an incentive system to ensure voting.
3. Supplying an easy way to submit proposals with a sorting mechanism to separate the valuable ballots from the irrational ones.

And to meet all these challenges in a distributed manner that doesn’t necessitate centralized governance.

A modularized treasury system compatible with most existing blockchains was designed and implemented by IOHK [Zhang_2019], supporting liquid democracy and delegative voting for better collaborative intelligence. Cardano has this adapted in its Catalyst system, which is currently live, but it will be more fully developed in the Voltaire era (5 of 5). The five eras of the Cardano platform are being developed synchronously, so some pieces are already available.

Privacy

Most BCs are public permissionless, which means that the contents of wallets and their transactions are available for anyone to view online freely. This lack of privacy is of concern to some individuals and much more so to organizations and businesses.

Several cryptocurrencies that take privacy as an imperative, including various degrees of anonymity: Monero, Dash, and Zcash. However, though these are not competitors to the BC 3.0 title, their privacy principles are being adapted into many contenders for the BC 3.0 title.

Adapted from the core concept of zero-knowledge proofs (ZKPs), zero-knowledge Succinct Non-interactive ARguments of Knowledge (zkSNARKs) and zk-Rollups offer a cryptographic privacy protocol proving ownership of a piece of knowledge without actually revealing it. For more, see Appendix A.

Ethereum 2.0 intends to use validity proofs in zk-rollups and Validium with data not stored on the main layer one.

Cardano offers Sonic, a zk-SNARK, supporting a universal and continually updatable structured reference string that scales linearly in size. Sonic proofs are constant in size, and cost of verification is comparable with the most efficient SNARKs in literature [Maller_2019]. As Cardano's SC capabilities (Goguen era: 3 of 5) will be released in the testnet stage at the end of April to early May 2021, Sonic must wait to be implemented.

Governance

The fifth and final stage of Cardano, the Voltaire era, will bring on-chain governance. Catalyst is one part of the Voltaire era that is already operational, though Catalyst is mainly about treasury, mentioned previously, providing finances to build on Cardano. Voltaire is about creating governance for protocol updates, supplying the

capacity to maintain and improve Cardano in a decentralized manner. Cardano is intended to be in the hands of the community, prospective completion 2025. The voting system is aimed to shift from the current system of Plutocracy to a hybrid of the current system and proof of merit.

Polkadot provides a sophisticated governance system allowing it to evolve under the direction of the majority of its stakeholders. Changes are made to the network by combining votes of active token holders and the council, a fixed number of seats (expected to increase to 24). Anyone can propose a referendum to the council by depositing a minimum number of tokens to support it. A proposal can be seconded by an agreeing member depositing the same number of tokens. The proposal with the highest amount of bonded tokens will be selected for inclusion as a referendum in the next voting cycle. There can only be a maximum of 100 public proposals in the proposal queue. There is also a council queue for proposals requiring a majority (51%) approval to be included as a referendum, with council members elected for the term of one week by token holders. One active referendum can be in progress at any given time, excluding the case of an emergency referendum - an accelerated proposal by the council or technical committee. The referendum is chosen as the top proposal, backed by the most bonded stake, and alternates between the two queues [Governance-Polkadot_wiki].

Table 2: Blockchain 3.0 works in progress

	Ethereum 2.0	Cardano	Zilliqa	Polkadot
MoC	PoS	PoS	pBFT	Hybrid PoS
Scalability	sharding, layer 2	sharding, layer 2	sharding, linear scaling	bonded parachains
Inter-operability	--	KEVM, Plutus, and IELE for correct by design SCs	--	cross-chain transfers of data or assets between parachains
Sustainability	--	Treasury system	--	--
Privacy	zk-rollups	Sonic zk-SNARK	--	bridges to zk-SNARK chains
Governance	--	democratic votes on proposals by stakeholders funded by the treasury system	--	democratic votes on proposals by stakeholders or council
Extra info	first mover advantage with SCs	academically peer reviewed	sharding at origin	designed for blockchain interoperability

Chapter 3 Key Takeaways

- Blockchain 1.0: cryptocurrency offering decentralized P2P transactions without a trusted third party.
- Blockchain 2.0: smart contracts are embedded terms and conditions via a programming language into cryptocurrency transactions.
- Blockchain 3.0: work in progress solving scalability, interoperability, sustainability, privacy, and governance for mainstream use of BC.
- Ethereum's EVM set the BC 2.0 stage with smart contracts growing more complex: Dapps, DAOs, DACs, and DASs.
- Smart contracts in Ethereum have possible bugs: time order dependent contracts, timestamp dependent, mishandled exceptions, and reentrancy vulnerability.
- Deterministic contracts are being developed: Ethereum 2.0's Python-like Vyper language; Cardano's compiler Plutus, for on- and off-chain smart contracts written in Haskell - a purely functional programming language.
- Scalability or transactions per second (tps) is a significant hindrance. Bitcoin processes 7tps and Ethereum can process 20tps while PayPal handles 200tps and VISA 20,000. Solutions include sharding, layer two fixes, and use of DAG instead of BC.
- Interoperability between BCs is necessary for mainstream adoption.
- Sustainability for ongoing funding of further development is necessary to prevent a BC from becoming obsolete.

- Privacy is vital as most BCs are public, permissionless, meaning all transactions and wallet amounts are visible to anyone. Zero-knowledge proofs (ZKP) prove ownership of a piece of knowledge without disclosing the knowledge itself.
- Governance allows a decentralized BC to be controlled via the democratic direction of stakeholders rather than a centralized authority.

4 - Self-Sovereign Identities

Identity is an inherent human condition; it is not conditional on outside administration. Self-sovereign identity is based upon the following principles [Allen_2016, Tobin_2016]:

Security the identity information must be kept secure	**Controllability** the user must be in control of who can see and access their data	**Portability** the user must be able to use their identity data whenever they want and not be tied to a single provider
Protection	Existence	Interoperability
Persistence	Existence	Interoperability
Minimization	Consent	Access

In the connected world of the Internet, digital identity has progressed from a centralized authority (e.g., IANA, ICANN) to administrative control by multiple federated authorities (e.g., Microsoft Passport, Liberty Alliance, Facebook Login), and on to user-centric individual control (e.g., OpenID, OAuth, FIDO). The concept of a self-sovereign identity is now coming into attention, requiring that the user is the ruler of their own identity [Allen_2016]. With the recent refugee crisis, resulting in multiple people without a recognized identity, and the European Union's General Data Protection Regulation's (GDPR) attempt to regulate customer data and privacy abuses, the need for a digital Self-Sovereign Identifier (SSI) is emphasized. For more regarding GDPR and its relation to BC, see Appendix B.

The current system of siloed identities and passwords has existed since the Internet was first built as "the Internet was created without an identity layer [Cameron_2005]." A much-needed evolution of the internet is creating a common identity layer allowing people, organizations, and things to have their own discrete SSI. This SSI ecosystem can now be built on top of DLT.

The premise of SSI is that it is an identity wholly owned by the individual without authorization or management by a central entity. To use an SSI, the holder presents claims about their identity that can be cryptographically verified by the verifier that desires to check them [Govind_2020].

The future of an active SSI ecosystem will include changes across the Internet. Most notably, there will be changes in how Big Data is collected and whether it will retain its ability to be collected legally or otherwise – e.g., through privacy-preserving zero-knowledge-proofs (ZKP). Investigation and documentation of changes and potential workarounds for big data collection are needed to anticipate when SSI use becomes standard across the web.

SSIs are sometimes referred to as Decentralized Identities. An SSI includes the following three components: Decentralized Identifiers (DIDs), Verifiable Credentials (VCs), and Identity Agents.

4.1. Decentralized Identifiers

The W3C is outlining DIDs as a new type of identifier that enables verifiable decentralized digital identities [Reed_2021]. A DID controller decides the subject that the DID identifies (e.g., a person, organization, thing, data model, etc.). DIDs are designed to enable individuals and organizations to create and prove control over their own identifiers, rather than relying on management by centralized authorities.

A DID is a globally unique URI to an identity: a web-accessible text string that resolves to a DID document (DIDdoc). The DID is a text string composed of three parts: the ‘did’ URI scheme identifier, the DID method’s identifier, and the DID method-specific identifier. Some DID method specifications currently in development can be found at [Steele_2020].

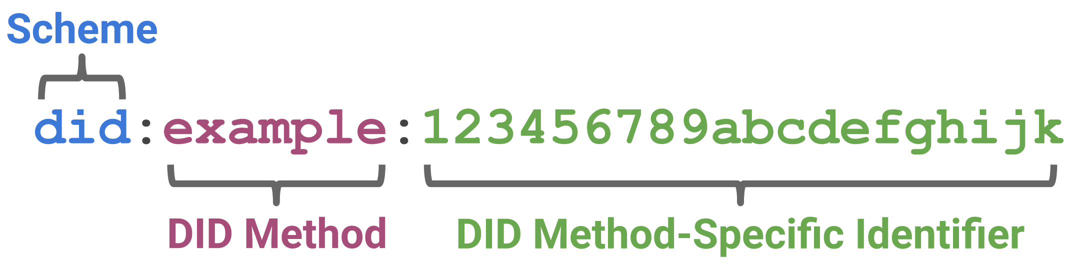

Scheme
did:example:123456789abcdefghijk
DID Method
DID Method-Specific Identifier

Figure 3: Example of a DID [Reed_2021]

The DIDdoc that is resolved with the DID contains the ways to cryptographically authenticate a DID controller, a set of data describing the DID subject, and other information associated with the DID. DIDdocs are most often in the form JSON or JSON-LD. However, developers are permitted to use other forms, such as XML or YAML. In Example 1, under authentication, the encryption type is specified along with the public key to verify the DID.

A DID is the digital equivalent of your state ID number

Example 1: A minimal DID document [Reed_2021]

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "ed25519verificationkey2020",
    "controller": "did:example:123456789abcdefghi",
    "publickeymultibase": "zh3c2avvlmv6gmmnam3uvajzpfkcjcwdwnzn6z3wxmqpv"
  }]
}
```

4.2 - Verifiable Credentials

VCs are a means to express credentials on the Web in a cryptographically secure and machine-verifiable way while respecting privacy. VCs are the digital equivalents of paper credentials like driver's licenses, government-issued IDs, and university degrees [Sporny_2019].

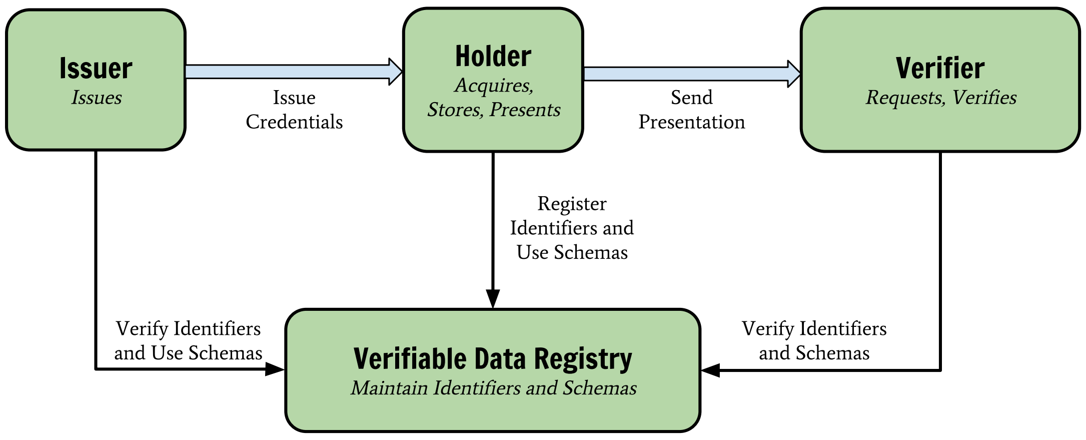

Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verify Identifiers and Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas

Figure 4: The roles and information flows forming a Verifiable Credential [Sporny_2019]

A VC is a tamper-proof credential with the issuer's authorship, enabling cryptographic verifiability, given to a holder, data subject. The set of claims in a VC can be used to build verifiable presentations, which are also cryptographically verified, to be shared with a verifier.

A Claim is expressed using a subject-property-value relationship about a subject used to express a large variety of statements. For example, a claim can be expressed where someone graduated from a particular university, see Fig. 5. Claims from multiple VCs can be combined in a single verifiable presentation.

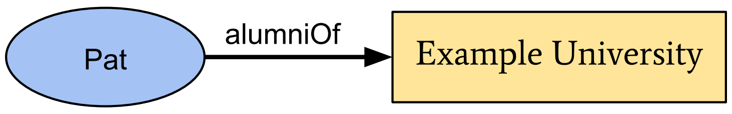


Figure 5: A basic claim expressing that Pat is an alumni of "Example University" [Sporny_2019]

Some verifiable presentations may contain data in synthesized form, such as for ZKP, rather than the original VC claims.

A Credential is a set of one or more claims made by the same entity (Issuer) and may contain an identifier and other metadata describing the credential's properties.

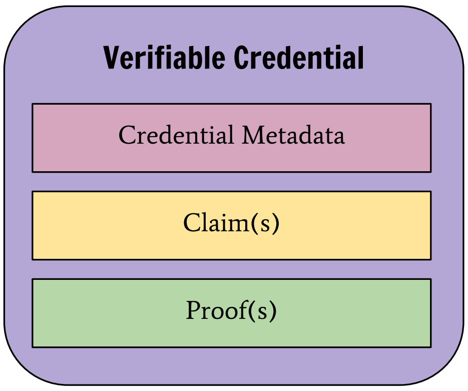


Figure 6: Components of a Verifiable Credential [Sporny_2019]

A VC is another JSON document that is stored in an Identity Agent:

Example 2: Example of a verifiable credential [Sporny_2019]

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/1872",
  "type": ["VerifiableCredential", "AlumniCredential"],
  "issuer": "https://example.edu/issuers/565049",
  "issuanceDate": "2010-01-01T19:73:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "alumniOf": {
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      "name": [{
        "value": "Example University",
        "lang": "en"
      }, {
        "value": "Exemple d'Université",
        "lang": "fr"
      }]
    }
  },
  "proof": {
    "type": "RsaSignature2018",
    "created": "2017-06-18T21:19:10Z",
    "proofPurpose": "assertionMethod",
"verificationMethod": "https://example.edu/issuers/keys/1",
"jws":"eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5XsITJX1CxPCT8yAVTV
kIEq_PbChOMqsLfRoPsnsgw5WEuts01mqQy7UJiN5mgRxDWUcX16dUEMGlv50aqzpqh4Qktb3rkBuQy72IFLOqV0
G_zS245kronKb78cPN25DGlcTwLtjPAYuNzVBAh4vGHSrQyHUdBBPM"
  }
}
```

The digital proof attached to the VC allows presenting a verifiable presentation to a verifier, attaching a digital signature at the end. See Example 3.

Example 3: Example of a verifiable presentation [Sporny_2019]

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "type": "VerifiablePresentation",

  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "id": "http://example.edu/credentials/1872",
    "type": ["VerifiableCredential", "AlumniCredential"],
    "issuer": "https://example.edu/issuers/565049",
    "issuanceDate": "2010-01-01T19:73:24Z",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "alumniOf": {
        "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
        "name": [{
          "value": "Example University",
          "lang": "en"
        }, {
          "value": "Exemple d'Université",
          "lang": "fr"
        }]
      }
    },
    "proof": {
      "type": "RsaSignature2018",
      "created": "2017-06-18T21:19:10Z",
      "proofPurpose": "assertionMethod",
      "verificationMethod": "https://example.edu/issuers/keys/1",
      "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5XsITJX1CxPC
      T8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-pQy7UJiN5mgRxD-WUcX16dUEMGlv50aqzpqh4Qkt
      b3rk-BuQy72IFLOqV0G_zS245-kronKb78cPN25DGlcTwLtjPAYuNzVBAh4vGHSrQyHUdBBPM"
    }
  }],

  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-09-14T21:19:10Z",
    "proofPurpose": "authentication",
    "verificationMethod": "did:example:ebfeb1f712ebc6f1c276e12ec21#keys-1",

    "challenge": "1f44d55f-f161-4938-a659-f8026467f126",
    "domain": "4jt78h47fh47",
    "jws": "eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..kTCYt5XsITJX1CxPCT
    8yAV-TVIw5WEuts01mq-pQy7UJiN5mgREEMGlv50aqzpqh4Qq_PbChOMqsLfRoPsnsgxD-WUcX16dUOqV0G_
    zS245-kronKb78cPktb3rk-BuQy72IFLN25DYuNzVBAh4vGHSrQyHUGlcTwLtjPAnKb78"
  }
}
```

4.3 - Identity Agents

Identity Agents are the interfaces (e.g., mobile wallets) that humans will use to interact with their identity, offering an interface to manage DIDs and VCs. However,

there are no fully working examples.

Examples under development:

- Microsoft Authenticator: already existing app from Microsoft that will be upgraded to support DID/VC use.
- Atala Prism: A demo app made by IOHK/IOG is available on the App Store and Google Play - more in Section 5.2.
- Hyperledger Aries: client-side components working together with Hyperledger Ursa for the cryptographic library and Hyperledger Indy (or another resolver) for distributed ledger purposes.

Chapter 4 Key Takeaways

- In the connected world of the Internet, digital identity has progressed from a centralized authority to federated authorities and user-centric individual control. The Self-Sovereign Identity (SSI) ecosystem is being created for individuals' full identity ownership without centralized authorization or control.
- An SSI (aka Decentralized Identities) is comprised of three components: Decentralized Identifiers (DIDs), Verifiable Credentials (VCs), and Identity Agents.
- The W3C is standardizing DIDs as a globally unique URI for an identity that resolves to a DID document containing ways to authenticate the DID controller cryptographically.
- VCs, also being standardized by the W3C, are a cryptographically secure, machine-verifiable set of claims about the holder.
- Claims from multiple VCs can be combined into a verifiable presentation to be presented to a verifier for cryptographic verification.
- Identity Agents are interfaces (e.g., mobile wallets) that humans use to interact with their identity and manage DIDs and VCs.

5 - SSI Ecosystem - Work in Progress

SSI work is under active development by multiple parties and collectives. These include Rebooting the Web of Trust (RWOT), Internet Identity Workshop (IIW), the World Wide Web Consortium (W3C), Sovrin, the Linux Foundation's Hyperledger Project, the Decentralized Identity Foundation (DIF), Atala Prism by IOG (formerly known as IOHK), and others. Many participants and entities have added pieces to the construction of the SSI ecosystem.

The SSI infrastructure has been formulating since 2015, but cryptography principles in the "Web of Trust" predated with Phil Zimmerman's PGP 2.0 back in 1992. Through the use of BC and other DLT, we have decentralized, immutable timestamps attached to public and private key pairs, enabling the infrastructure on which to build SSI for use across the globe.

5.1 - Hyperledger Identity Stack: Indy, Aries, ad Ursa

The Linux Foundation set up the Hyperledger Project in 2015 to advance cross-industry collaboration in developing the performance and reliability of blockchains and distributed ledgers for non-cryptocurrency use. Hyperledger Indy (Indy) joined in 2017 as the first identity-focused blockchain framework. Indy's code was contributed by the Sovrin Foundation, a global non-profit intending to provide a new standard for digital identities.

As Indy evolved, in 2018, the cryptography repository was migrated into its own project, Hyperledger Ursa (Ursa), to enable easier integration into other Hyperledger projects. Hyperledger Aries (Aries) was added in 2019 as a toolkit providing interoperability of agents using DIDs and VCs from multiple ecosystems. The Hyperledger Identity stack is a combination of Hyperledger Aries, where most developers will build on, on top of Hyperledger Indy, as an identity ledger. at the bottom are the cryptographic elements of Hyperledger Ursa.

The Sovrin Network is a single global instance of Indy with nodes operated by a Sovrin Steward, an organization (company, government, university, etc.) that has made a legal agreement defining how they will operate their node within the rules defined by the Sovrin governance framework. The Sorvrin Foundation provides governance for the network, formulating the rules and upgrades of the nodes and overseeing the business and legal aspects of the network.

Endorsers in the Sovrin Network are entrusted through reputation, legal agreements, and adherence to the Sovrin governance framework. Those they 'anoint' as other trusted known identities are solely granted permission to write to the public ledger. Indy was designed as a public permissioned ledger: anyone can read, but only a known few can write.

As Indy is public, all data that is written to the BC is not private. Therefore, no individual's VCs, DIDs, or other private data are permitted to be recorded to the ledger. Only credential issuers (e.g., governments, schools, etc.) need to write two, three, or four pieces of data to the ledger for a nominal cost [Sovrin_2020]:

- **Public DIDs:** An issuer of a credential MUST have a DID recorded on the BC to allow verifiers to know who they are and why the credentials they issued can be trusted.
- **Schemas:** A list of attribute names (fields) and their data types that a credential will have. A credential Issuer MAY put a new schema or use one that others already placed on the.

- **Credential Definitions:** The DID of the issuer, the schema for the credentials, and the public keys (one per claim) used to sign the claims to issue a credential MUST be written to the BC.
- **Revocation Registry:** The issuer MAY require the ability to revoke credentials, and if so, they must write a revocation registry to the ledger before issuing credentials. The revocation registry links to the credential definition, allowing the issuer to revoke credentials independent of the holders. The revoked credentials are not listed for public visibility, rather using ZKP, a holder can prove, and a verifier can verify that the holder's unrevoked credential has not been revoked.

Aries is meant to be BC agnostic, and as mentioned above, Aries is what most developers will build on. An Aries agent is a piece of software that handles VCs for an entity (person, organization, or thing), allowing the holder to receive VCs from issuers and provide verifiable presentations to verifiers – a digital wallet. Agents use private, pairwise DIDs for every relationship, providing end-to-end encryption, allowing certain knowledge of who is on the other end of the connection.

Types of agents:

- **Personal Agents/Edge Agents:** a mobile or PC app to establish connections and exchange messages such as offers of credentials and requests to prove claims from credentials. Personal agents could also be operated and run by a cloud
  service, but this would require trust in the service as it would have control of all keys.
- **Enterprise Agents:** run on servers to verify claims and issue credentials to clients. An enterprise will use VCs for received licenses and permits.
- **Device Agents:** IoT devices can put sensor data into VCs to prevent tampering.
- **Routing Agent/Cloud Agents:** serve as intermediaries to facilitate the flow of encrypted messages between other types of agents.

Aries agents are divided into the following components: KMS, messaging interface, ledger interface, and controller.

An Aries agent has secure storage called the key management service (KMS) to store all information collected by the agent. An Aries KMS is wrapper code around a database (e.g., SQLite, PostgreSQL) that contains DIDs, keys, credentials, etc., that are stored encrypted. The KMS key pair encrypts all other keys and data, and the KMS access key is protected by the app platform's capable means (e.g., fingerprint, facial recognition, etc.). For backup purposes, not only must the database be backed up, but the keys to decrypt the database entries must be backed up. These must be packaged, encrypted, and protected by a recovery key pair and must be held safely, separately from the backup until it is needed to restore. For possible ways to manage backup and recovery key storage, see Appendix 2.

Aries uses transport-agnostic DID Communication (DIDComm) messaging protocols to enable P2P messaging. The message is encrypted via Ursa and assembled into a JSON Web Encryption (JWE) structure by the DIDComm envelope protocol and decrypted at the other end by the intended recipient.

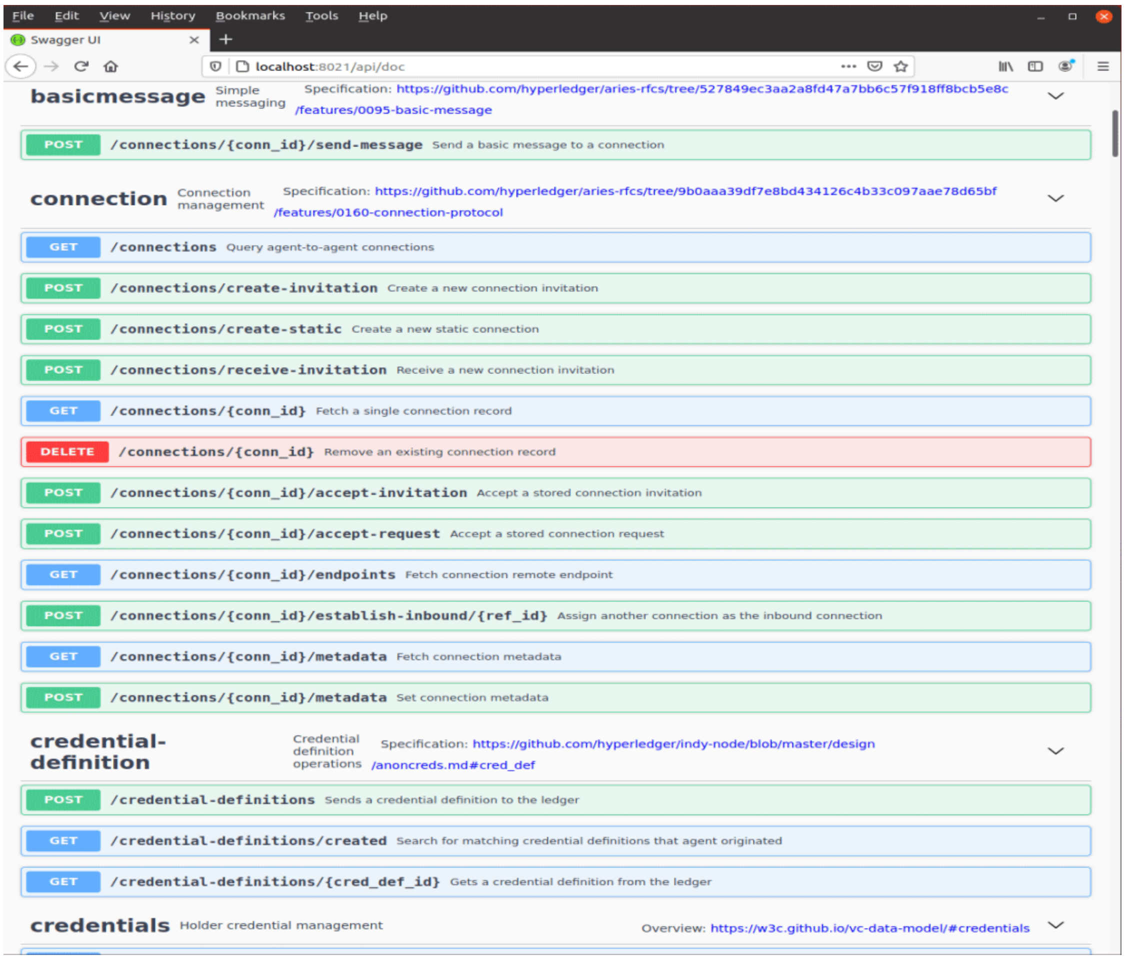
The Aries ledger interface is pluggable, allowing an Aries agent to interact with multiple ledgers and VC ecosystems. However, Indy is the only ledger interface implementation currently.

The controller provides the rules defining what actions the agent initiates and how it will respond to events. In a mobile agent, the controller is a user interface presenting the options to a person. An issuer/verifier enterprise agent might be a legacy database system that manages consumer data, potentially enabling customers to submit data by presenting claims rather than typing the information in a web form.

For development, an Aries Agent can be divided into two logical components: framework and controller. The framework corresponds to the standard capabilities provided to enable the Aries agent to interact with its surroundings: ledgers, storage, and other agents. The controller is the customizable component designed to handle rules for that particular agent.

Aries frameworks for edge development are currently available in .NET [Hyperledger_Aries-framework-dotnet] and golang [Hyperledger_Aries-framework-go], and there is work on Java and JavaScript based. For cloud development that is not designed for mobile devices is available in Python: Aries Cloud Agent Python [Hyperledger_ACA-Py]. The frameworks for edge development are embedded in the controller enabling the use of DIDComm and other Aries protocols.

Aries Cloud Agent Python can be accessed through an Open-API/Swagger UI, allowing manual simulation of a connection between Faber college and Alice [Hyperledger_AriesOpenAPIDemo]. The ledger used in this demo is the GreenLight Dev Ledger, an open public ledger contributed and operated by the Province of British Columbia [Greenlight]. The protocols for connection can be seen in Fig. 7, followed by Faber sending Alice an invitation in Fig. 8 and Alice receiving the invitation in Fig. 9.



basicmessage Simple messaging Specification: https://github.com/hyperledger/aries-rfcs/tree/527849ec3aa2a8fd47a7bb6c57f918ff8bcb5e8c/features/0095-basic-message

POST /connections/{conn_id}/send-message Send a basic message to a connection

connection Connection management Specification: https://github.com/hyperledger/aries-rfcs/tree/9b0aaa39df7e8bd434126c4b33c097aae78d65bf/features/0160-connection-protocol

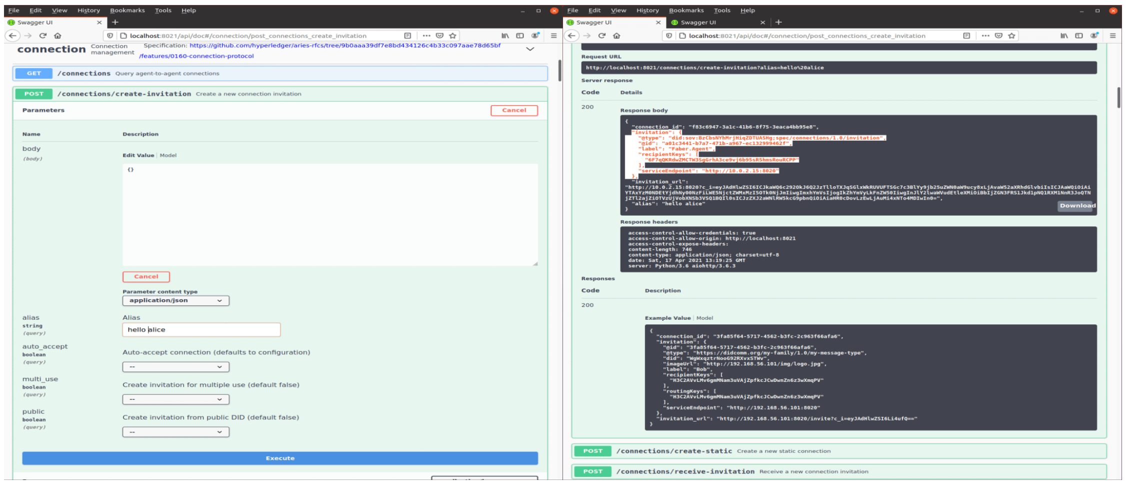
GET /connections Query agent-to-agent connections
POST /connections/create-invitation Create a new connection invitation
POST /connections/create-static Create a new static connection
POST /connections/receive-invitation Receive a new connection invitation
GET /connections/{conn_id} Fetch a single connection record
DELETE /connections/{conn_id} Remove an existing connection record
POST /connections/{conn_id}/accept-invitation Accept a stored connection invitation
POST /connections/{conn_id}/accept-request Accept a stored connection request
GET /connections/{conn_id}/endpoints Fetch connection remote endpoint
POST /connections/{conn_id}/establish-inbound/{ref_id} Assign another connection as the inbound connection
GET /connections/{conn_id}/metadata Fetch connection metadata
POST /connections/{conn_id}/metadata Set connection metadata

credential-definition Credential definition operations Specification: https://github.com/hyperledger/indy-node/blob/master/design/anoncreds.md#cred_def

POST /credential-definitions Sends a credential definition to the ledger
GET /credential-definitions/created Search for matching credential definitions that agent originated
GET /credential-definitions/{cred_def_id} Gets a credential definition from the ledger

credentials Holder credential management Overview: https://w3c.github.io/vc-data-model/#credentials

Figure 7: Swagger UI Connection management


File Edit View History Bookmarks Tools Help
Swagger UI
connection Connection management
GET /connections Query agent-to-agent connections
POST /connections/create-invitation Create a new connection invitation
Parameters
Cancel
Name
Description
Edit Value | Model
Parameter content type
Execute
Request URL
Server response
Code
Details
Response body
Response headers
Responses
Description
Example Value | Model
Download
POST /connections/create-static Create a new static connection
POST /connections/receive-invitation Receive a new connection invitation

```
localhost:8021/api/doc#/connection/post_connections_create_invitation
Specification: https://github.com/hyperledger/aries-rfcs/tree/9b0aaa39df7e8bd434126c4b33c097aae78d65bf/features/0160-connection-protocol

body (body)       {}
Parameter content type: application/json
alias string (query)          Alias: hello alice
auto_accept boolean (query)   Auto-accept connection (defaults to configuration): --
multi_use boolean (query)     Create invitation for multiple use (default false): --
public boolean (query)        Create invitation from public DID (default false): --
```

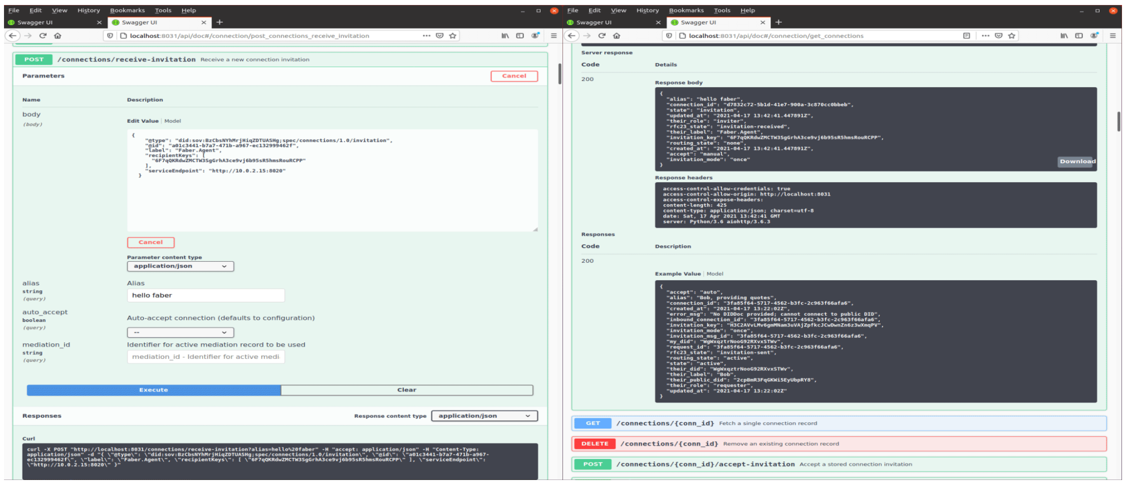
```
Request URL
http://localhost:8021/connections/create-invitation?alias=hello%20alice

Server response
Code  Details
200   Response body
{
  "connection_id": "f83c6947-3a1c-41b6-8f75-3eaca4bb95e8",
  "invitation": {
    "@type": "did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/connections/1.0/invitation",
    "@id": "a01c3441-b7a7-471b-a967-ec132999462f",
    "label": "Faber.Agent",
    "recipientKeys": [
      "6F7qQKRdwZMCTW3SgGrhA3ce9vj6b95sR5hmsRouRCPP"
    ],
    "serviceEndpoint": "http://10.0.2.15:8020"
  },
  "invitation_url":
"http://10.0.2.15:8020?c_i=eyJAdHlwZSI6ICJkaWQ6c292OkJ6Q2JzTlloTXJqSGlxWkRUVUFTSGc7c3BlYy9jb25uZWN0aW9ucy8xLjAvaW52aXRhdGlvbiIsICJAaWQiOiAi
YTAxYzM0NDEtYjdhNy00NzFiLWE5NjctZWMxMzI5OTk0NjJmIiwgImxhYmVsIjogIkZhYmVyLkFnZW50IiwgInJlY2lwaWVudEtleXMiOiBbIjZGN3FRS1Jkd1pNQ1RXM1NnR3JoQTN
jZTl2ajZiOTVzUjVobXNSb3VSQ1BQIl0sICJzZXJ2aWNlRW5kcG9pbnQiOiAiaHR0cDovLzEwLjAuMi4xNTo4MDIwIn0=",
  "alias": "hello alice"
}

Response headers
access-control-allow-credentials: true
access-control-allow-origin: http://localhost:8021
access-control-expose-headers:
content-length: 746
content-type: application/json; charset=utf-8
date: Sat, 17 Apr 2021 13:19:25 GMT
server: Python/3.6 aiohttp/3.6.3

Responses
Code  Description
200   Example Value
{
  "connection_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "invitation": {
    "@id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "@type": "https://didcomm.org/my-family/1.0/my-message-type",
    "did": "WgWxqztrNooG92RXvxSTWv",
    "imageUrl": "http://192.168.56.101/img/logo.jpg",
    "label": "Bob",
    "recipientKeys": [
      "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    ],
    "routingKeys": [
      "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    ],
    "serviceEndpoint": "http://192.168.56.101:8020"
  },
  "invitation_url": "http://192.168.56.101:8020/invite?c_i=eyJAdHlwZSI6Li4ufQ=="
}
```

Figure 8: Faber create-invitation demo


File Edit View History Bookmarks Tools Help
Swagger UI
POST /connections/receive-invitation Receive a new connection invitation
Parameters
Cancel
Name
Description
Edit Value | Model
Execute
Clear
Responses
Response content type
Curl
Server response
Code
Details
Response body
Response headers
Description
Example Value | Model
Download
GET /connections/{conn_id} Fetch a single connection record
DELETE /connections/{conn_id} Remove an existing connection record
POST /connections/{conn_id}/accept-invitation Accept a stored connection invitation

```
localhost:8031/api/doc#/connection/post_connections_receive_invitation

body (body)
{
    "@type": "did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/connections/1.0/invitation",
    "@id": "a01c3441-b7a7-471b-a967-ec132999462f",
    "label": "Faber.Agent",
    "recipientKeys": [
      "6F7qQKRdwZMCTW3SgGrhA3ce9vj6b95sR5hmsRouRCPP"
    ],
    "serviceEndpoint": "http://10.0.2.15:8020"
  }
Parameter content type: application/json
alias string (query)           Alias: hello faber
auto_accept boolean (query)    Auto-accept connection (defaults to configuration): --
mediation_id string (query)    Identifier for active mediation record to be used: mediation_id - Identifier for active medi

Response content type: application/json
Curl
curl -X POST "http://localhost:8031/connections/receive-invitation?alias=hello%20faber" -H "accept: application/json" -H "Content-Type:
application/json" -d "{ \"@type\": \"did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/connections/1.0/invitation\", \"@id\": \"a01c3441-b7a7-471b-a967-
ec132999462f\", \"label\": \"Faber.Agent\", \"recipientKeys\": [ \"6F7qQKRdwZMCTW3SgGrhA3ce9vj6b95sR5hmsRouRCPP\" ], \"serviceEndpoint\":
\"http://10.0.2.15:8020\" }"
```

```
localhost:8031/api/doc#/connection/get_connections

Server response
Code  Details
200   Response body
{
  "alias": "hello faber",
  "connection_id": "d7832c72-5b1d-41e7-900a-3c870cc0bbeb",
  "state": "invitation",
  "updated_at": "2021-04-17 13:42:41.447891Z",
  "their_role": "inviter",
  "rfc23_state": "invitation-received",
  "their_label": "Faber.Agent",
  "invitation_key": "6F7qQKRdwZMCTW3SgGrhA3ce9vj6b95sR5hmsRouRCPP",
  "routing_state": "none",
  "created_at": "2021-04-17 13:42:41.447891Z",
  "accept": "manual",
  "invitation_mode": "once"
}

Response headers
access-control-allow-credentials: true
access-control-allow-origin: http://localhost:8031
access-control-expose-headers:
content-length: 425
content-type: application/json; charset=utf-8
date: Sat, 17 Apr 2021 13:42:41 GMT
server: Python/3.6 aiohttp/3.6.3

Responses
Code  Description
200   Example Value
{
  "accept": "auto",
  "alias": "Bob, providing quotes",
  "connection_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "created_at": "2021-04-17 13:22:02Z",
  "error_msg": "No DIDDoc provided; cannot connect to public DID",
  "inbound_connection_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "invitation_key": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV",
  "invitation_mode": "once",
  "invitation_msg_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "my_did": "WgWxqztrNooG92RXvxSTWv",
  "request_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "rfc23_state": "invitation-sent",
  "routing_state": "active",
  "state": "active",
  "their_did": "WgWxqztrNooG92RXvxSTWv",
  "their_label": "Bob",
  "their_public_did": "2cpBmR3FqGKWi5EyUbpRY8",
  "their_role": "requester",
  "updated_at": "2021-04-17 13:22:02Z"
}
```

Figure 9: Alice receive-invitation demo

After Alice receives the invitation, she will need to accept the invitation, send a corresponding create invitation back to Faber, which Faber must accept to have a connection established. After a connection is established, credentials can be issued and received, and private messages can be sent between contacts. This is an example demonstration of Aries protocols being developed that the SSI ecosystem will use.

Hyperledger provides a GitHub repository for Aries RFCs documenting the latest concepts and features [Hyperledger_Aries-RFCs]. Links to all RFC updates will remain though control of some of the DIDComm standards is moving to the DIF.

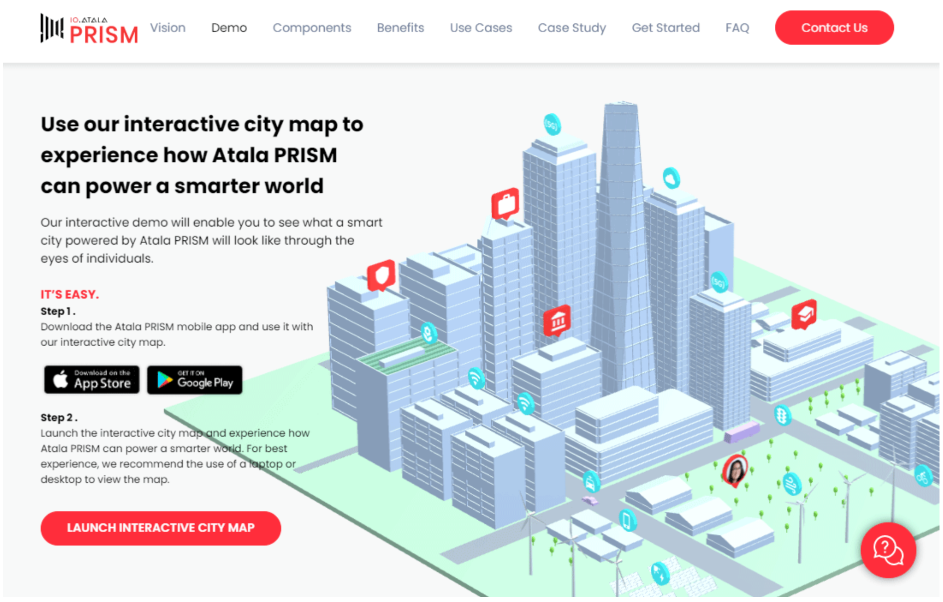
5.2 - Atala Prism

Atala, an enterprise BC framework similar to Hyperledger Fabric used by IBM BC, was developed by IOHK to onboard governments in developing countries. Atala was built to handle real-world use cases such as property registration, voting systems, and supply chain management, focusing on digital currency adoption. Starting with Addis Ababa, Ethiopia’s capital, MoUs have been signed to create a digital payment system, allowing six million users to pay power and electricity bills with cryptocurrency, eventually combining this system with an identity card [Wolfson_2019]. In the Republic of Georgia, Atala is teaming up with the Ministry of Education and universities to enable students to receive, store and send their achievements from their phones. This eliminates the need for background checks and saves time for universities and employers.

As enterprises and governments want more control over data for legal and regulatory reasons, Atala was developed as a permissioned system based on Cardano. Cardano, in its initial era (Byron era), was a permissioned system. Currently, Atala focuses on layer two solutions such as Prism for SSI and economic identity, Hydra for scaling, and more. These layer two solutions are blockchain agnostic, and as such, can be used on both permissioned (Atala) and permissionless (Cardano). Since Atala and Cardano are very similar, when Cardano improves, Atala will also improve through downstream contributions such as the EUTXO model and Plutus SCs. IOHK is working with the Hyperledger Foundation to onboard Gerolamo, a unification of permissioned and permissionless chains for fluid portability between enterprise and public usage of the Cardano BC [Hoskinson_2020].

The Cardano and Atala teams aim to provide the technology for digital identity to bank the unbanked and offer the means for developing countries to leapfrog legacy systems directly to the integrated digital world. In banking the unbanked and giving them access to a cryptocurrency for authenticating the supply chain, there comes an avenue to supply loans to local farmers and others.

Atala Prism is still under development, and the code has not yet been released open-source [IOHK_January2021]. The following is the available demonstration part on web, part on a mobile device [IOHK_AtalaPrism].



Use our interactive city map to experience how Atala PRISM can power a smarter world

Our interactive demo will enable you to see what a smart city powered by Atala PRISM will look like through the eyes of individuals.

IT'S EASY.

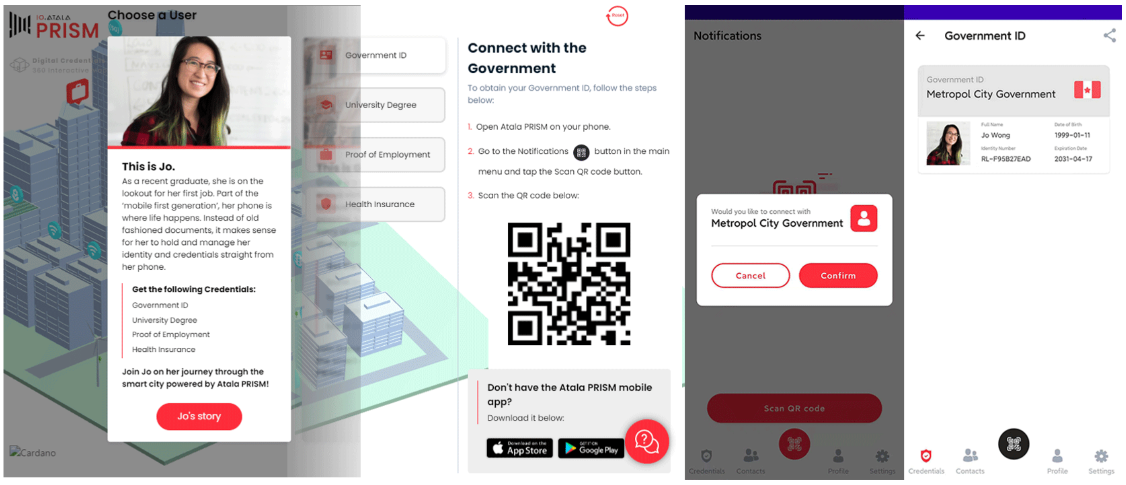
Step 1.
Download the Atala PRISM mobile app and use it with our interactive city map.

Step 2.
Launch the interactive city map and experience how Atala PRISM can power a smarter world. For best experience, we recommend the use of a laptop or desktop to view the map.

Figure 10: Atala Prism demo on www.atalaprism.io

When the mobile app is installed on a phone, there is an initial seed phrase of 12 words you are instructed to save. If the phone is lost or broken, it is the only way to recover the account. If the seed phrase is lost, all contacts and VCs will have to be collected from scratch.

The demonstration follows the process for Jo to get four credentials: Government ID, University Degree, Proof of Employment and Health Insurance. In the initial steps concatenated in Figure 11, a QR code is provided to be scanned by the mobile app Atala Prism. After scanning, on the mobile device is a confirmation to accept connection with the Metropol City Government. On confirmation, the Metropol City Government is added as a contact, and the Government ID in the far right of Figure 11 is available as a credential.



This is Jo.
As a recent graduate, she is on the lookout for her first job. Part of the 'mobile first generation', her phone is where life happens. Instead of old fashioned documents, it makes sense for her to hold and manage her identity and credentials straight from her phone.

Get the following Credentials:
Government ID
University Degree
Proof of Employment
Health Insurance

Join Jo on her journey through the smart city powered by Atala PRISM!

Connect with the Government
To obtain your Government ID, follow the steps below:
1. Open Atala PRISM on your phone.
2. Go to the Notifications button in the main menu and tap the Scan QR code button.
3. Scan the QR code below:

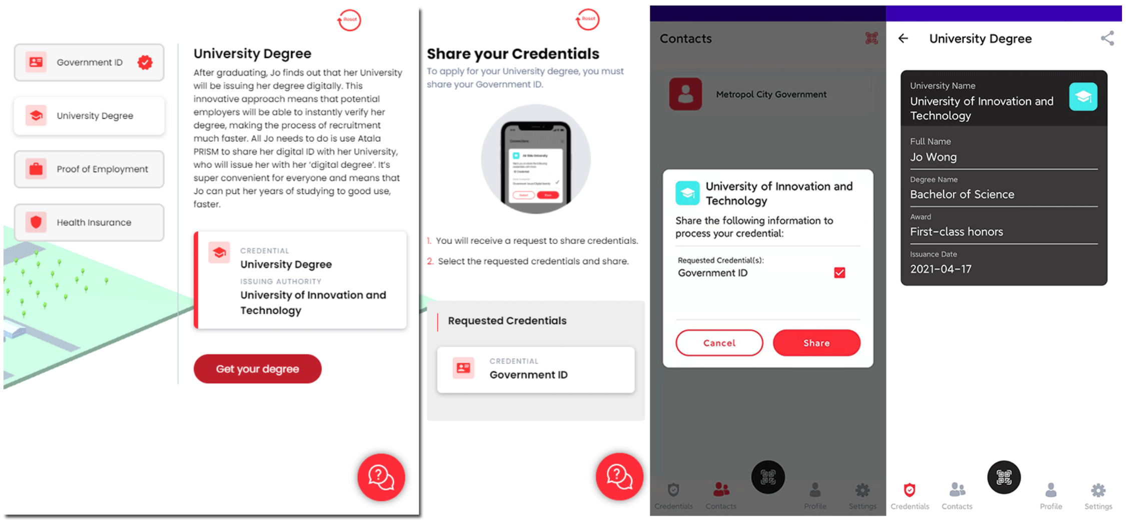
Don't have the Atala PRISM mobile app?
Download it below:

Would you like to connect with
Metropol City Government

Government ID
Government ID
Metropol City Government
Full Name: Jo Wong
Date of Birth: 1999-01-11
Identity Number: RL-F95B27EAD
Expiration Date: 2031-04-17

Figure 11: Atala Prism initial sequence for Jo to get a Government ID, for the illustrated Metropol City

Once the Government ID has been accepted, the process for getting degree credentials becomes available, as seen in Figure 12. In this process, the University requests the Government ID credential, which, when provided, adds the University as another contact and provides the University Degree as a credential on the mobile device.



University Degree
After graduating, Jo finds out that her University will be issuing her degree digitally. This innovative approach means that potential employers will be able to instantly verify her degree, making the process of recruitment much faster. All Jo needs to do is use Atala PRISM to share her digital ID with her University, who will issue her with her 'digital degree'. It's super convenient for everyone and means that Jo can put her years of studying to good use, faster.

CREDENTIAL
University Degree
ISSUING AUTHORITY
University of Innovation and Technology

Share your Credentials
To apply for your University degree, you must share your Government ID.
1. You will receive a request to share credentials.
2. Select the requested credentials and share.

Requested Credentials
CREDENTIAL
Government ID

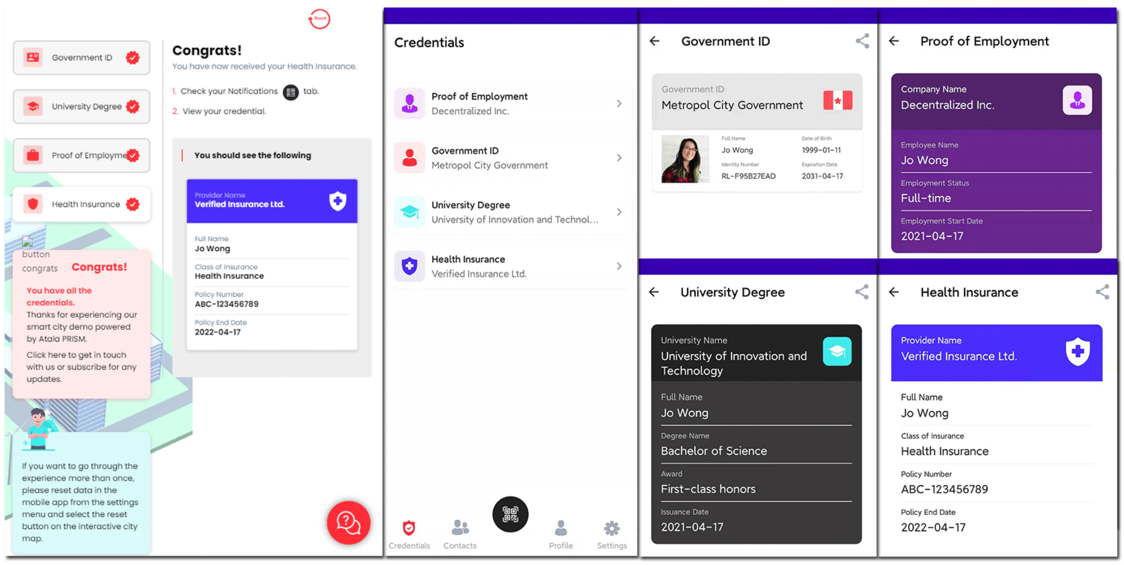
Metropol City Government

University of Innovation and Technology
Share the following information to process your credential:
Requested Credential(s):
Government ID ✓

University Degree
University Name: University of Innovation and Technology
Full Name: Jo Wong
Degree Name: Bachelor of Science
Award: First-class honors
Issuance Date: 2021-04-17

Figure 12: Atala Prism demo showing Jo applying for her University credential

A similar procedure is followed for Jo to get a job. In that step, both the Government ID and the University Degree credentials are requested. Once given, and presumably, Decentralized Inc. agrees to hire her, a Proof of Employment credential is provided. This Proof of Employment credential is then needed for the Health Insurance certificate. Figure 13 shows the final process with all four credentials received and shown on the far right.



Congrats!
You have now received your Health Insurance.
1. Check your Notifications tab.
2. View your credential.

You should see the following

Provider Name
Verified Insurance Ltd.
Full Name
Jo Wong
Class of Insurance
Health Insurance
Policy Number
ABC-123456789
Policy End Date
2022-04-17

Congrats!
You have all the credentials.
Thanks for experiencing our smart city demo powered by Atala PRISM.
Click here to get in touch with us or subscribe for any updates.

If you want to go through the experience more than once, please reset data in the mobile app from the settings menu and select the reset button on the interactive city map.

Credentials
Proof of Employment
Decentralized Inc.
Government ID
Metropol City Government
University Degree
University of Innovation and Technol...
Health Insurance
Verified Insurance Ltd.
Credentials Contacts Profile Settings

Government ID
Government ID
Metropol City Government
Full Name
Jo Wong
Date of Birth
1999-01-11
Identity Number
RL-F95B27EAD
Expiration Date
2031-04-17

Proof of Employment
Company Name
Decentralized Inc.
Employee Name
Jo Wong
Employment Status
Full-time
Employment Start Date
2021-04-17

University Degree
University Name
University of Innovation and Technology
Full Name
Jo Wong
Degree Name
Bachelor of Science
Award
First-class honors
Issuance Date
2021-04-17

Health Insurance
Provider Name
Verified Insurance Ltd.
Full Name
Jo Wong
Class of Insurance
Health Insurance
Policy Number
ABC-123456789
Policy End Date
2022-04-17

Figure 13: Atala Prism demo final showing or all four credentials for Jo

Anonymized threshold proofs are intended as the next level for Atala Prism, allowing specific characteristics to be proven without disclosing other data. This "where Prism is going as a product year 3 to year 5", says Hoskinson [Simmons_2021].

5.3 - Decentralized Identity Foundation

The DIF has been focused on the open-source development of specifications and standards for protocols, components, and data formats for industry-wide collaboration in decentralized identity / SSI construction. In the DIF, working groups collaborate on defining the standard specifications for the following areas [DIF]:

- **Identifiers and Discovery:** Universal Resolver & Registrar
- **Authentication:** DID-based authentication specs
- **Claims and Credentials:** DID Credential Manifest
- **DID Communication:** creating a standardized means of authenticated message passing P2P between DID controllers (furthering the work by Hyperledger Aries)
- **Sidetree:** blockchain agnostic, layer two, DID Method [DIF_Sidetree]
- **Secure Data Storage:** formulizing specifications for a foundational secure data storage layer

The majority of the content from the above working groups is standardization. However, one notable contribution comes in the form of a Universal Resolver (UR). The UR is a foundational piece of infrastructure for the SSI ecosystem by resolving the DIDdoc from a supplied DID [Sabadello_2021]. Currently, the development UR (https://dev.uniresolver.io) can support 42 DID Methods. This is not intended for more than development use, as being on a single server, the dev UR centralizes an intended decentralized infrastructure. When the SSI ecosystem is closer to final development, multiple servers will offer their own UR. The universal resolver protocol and appropriate drives are meant to be downloaded and run on more instances to maintain the distributed nature of the SSI ecosystem [DIF_Universal_Resolver].

Another prominent project from DIF is Sidetree, a BC agnostic layer two protocol for PKI. The Sidetree protocol defines state change operations (i.e., Create, Read, Update, or Deactivate) to mutate a DID's DIDdoc. Sidetree nodes anchor Content-Addressable Storage (CAS) references to the underlying anchoring system (e.g., DLT), offering an immutable chronology that all nodes can validate.

Building on Sidetree, Microsoft, a member of DIF, has been working on ION [DIF_ION]. ION uses Bitcoin as the underlying DLT and uses IPFS (see appendix 3) for CAS storage of DID operations (Fig. 14). Since ION nodes process transactions on layer two, it is possible to anchor tens of thousands of DID/DPKI operations in a single on-chain transaction. ION nodes can fetch, process, and assemble DID states in parallel, solving the scalability problem.

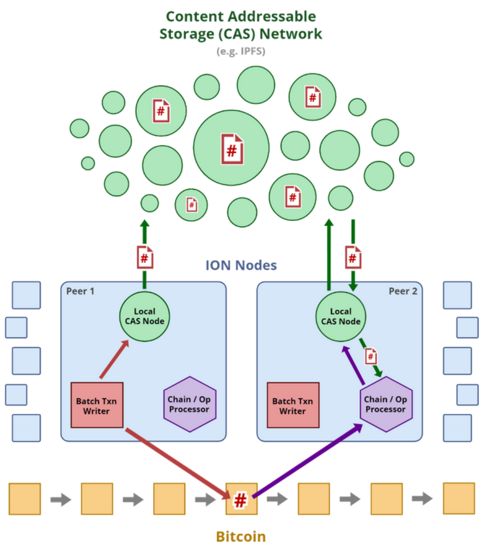


Figure 14: ION overview [source Microsoft techcommunity]

ION v1 is now complete and launched on Bitcoin mainnet, yet the process is still in the early phases. With v1, the following are possible [Dingle_2021]:

- Public preview of the Azure AD VC service
- OpenID Connect Self-Issued DID authentication with sites, apps, and services that implement the draft specification
- DID creation and cryptographic linking to controlled web domains for companies and individuals
- Use DIDs to issue VCs

Table 3: Comparison of SSI Infrastructure available

	ION	**Atala Prism**	**Hyperledger Aries**
Makers	DIF + Microsoft	IOG (formerly IOHK)	The Sovrin Foundation + Hyperledger (Linux Foundation)
DLT used	Bitcoin core	Cardano	Any resolver (blockchain interface layer. E.g., Hyperledger Indy)
DLT details	- Public - Permissionless - PoW	- Public - Permissionless - PoS	- Public - Permissioned - BFT
DPKI used	ION (Sidetree-based layer 2)	unknown	Hyperledger Indy
Open-source	yes (on git-hub)	yes (code not yet released)	yes (on git-hub)
Notes	Use of Azure for BaaS under construction.	Demo available on Apple App Store, and Google Play	Encourages using pairwise DIDs per relationship to prevent data correlation. ZKP from Hyperledger Ursa.

Chapter 5 Key Takeaways

- The Linux Foundation set up the Hyperledger project in 2015 to advance cross-industry collaboration developing blockchains and distributed ledgers for non-cryptocurrency use.
- Hyperledger Indy (distributed ledger), Ursa (cryptography), and Aries (agents) form the Hyperledger Identity stack, the first identity-focused blockchain framework.
- Indy is publicly readable with permissioned write access. It is intended for only credential issuers to write public DIDs, credential schemas and definitions, and optional revocation registries. No personally identifiable information is allowed.
- Aries is intended to be blockchain agnostic or interoperable.
- Aries agents are divided into KMS, messaging interface, ledger interface, and controller. or for development purposes: framework and controller.
- Aries frameworks for edge development are currently available in .NET and golang. For non-mobile development Aries Cloud Agent Python is available.
- Aries RFCs are available on a GitHub repository.
- Atala Prism is being developed by IOHK as an enterprise version of Cardano to offer SSI for governments in developing countries and beyond.
- A part mobile, part web demonstration is available and demonstrated here.
- The Decentralized Identity Foundation (DIF) is focused on the open-source development of standards for protocols, components, and data formats for industry-wide construction of decentralized identity/SSI.
- The DIF contributed a Universal Resolver (UR) to resolve the DIDdoc from a supplied DID.
- Building on the DIF project, Sidetree, Microsoft, a member of DIF, has launched V1 of ION. ION is a layer two DPKI on Bitcoin, the underlying DLT, and uses IPFS for content-addressable storage (CAS) of DID operations.

Conclusion

The SSI ecosystem being developed will bring significant changes to digital identity. Through the use of DLT, we have the technological infrastructure on which to build decentralized identities. On the one hand, decentralized identities place control back into the hands of the identity holders or controllers in the case of non-animate subjects, and on the other hand, allow verifiable traceability for identity data. Verifiable traceability improves big data quality from its collection in centralized, unverifiable islands to one that is straight from the source with consent receipts attached.

This paper presented the fundamental principles of Distributed Ledger Technology, Blockchain, Blockchain means of consensus, and blockchain's generations: cryptocurrency (BC 1.0), smart contracts (BC 2.0), and ongoing improvements in scalability, interoperability, sustainability, privacy, and governance (BC 3.0). This paper also described Self-Sovereign Identities, SSI subsections: Decentralized Identifiers, Verifiable Credentials, and Identity Agents, GDPR data protection regulations, and SSI work in progress: Hyperledger Indy, Ursa, and Aries, Atala Prism, and DIF member Microsoft's ION.

In future research, when the SSI infrastructure has developed further, exploration of uncorrelatable links to SSI claims through ZKP can be used to store knowledge of preferences in a globally accessible way (e.g., movie ratings categorized by age, gender, nationality, etc.).

References

[Allen_2016] Allen, Christopher. “The Path to Self-Sovereign Identity.” lifewithalacrity.com, April 2016. http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[Becker_2008] Becker, Georg. Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis, 2008.

[Benet_2014] Benet, Juan. “IPFS - Content Addressed, Versioned, P2P File System.” CoRR abs/1407.3561 (2014). http://arxiv.org/abs/1407.3561.

[Cameron_2005] Cameron, Deborah, and Don Kulick. “Identity Crisis?” Language & Communication 25, no. 2 (2005): 107–25. https://doi.org/10.1016/j.langcom.2005.02.003.

[Castro_1999] Castro, Miguel, and Barbara Liskov. “Practical Byzantine Fault Tolerance.” Proceedings of the Third Symposium on Operating Systems Design and Implementation 99 (February 1999): 173–86.

[Chakravarty_2020] Chakravarty, Manuel M. T., Sandro Coretti, Matthias Fitzi, Peter Gazi, Philipp Kant, Aggelos Kiayias, and Alexander Russell. Hydra: Fast Isomorphic State Channels, 2020.

[Chen_2019] Chun, Yun, Hui Xie, Kun Lv, Shengjun Wei, and Changzhen Hu. “DEPLEST: A Blockchain-Based Privacy-Preserving Distributed Database toward User Behaviors in Social Networks.” Information Sciences 501 (October 2019): 100–117. https://doi.org/10.1016/j.ins.2019.05.092.

[Cohen_2003] Cohen, Bram. Incentives Build Robustness in BitTorrent, 2003.

[Cui_2019] Cui, Jiaming, Huaying Wu, Luoyi Fu, and Xiaoying Gan. “De-Anonymizing Bitcoin Networks: An IP Matching Method via Heuristic Approach: Poster.” ACM TURC ’19. New York, NY, USA: Association for Computing Machinery, 2019. https://doi.org/10.1145/3321408.3321607.

[DIF] DIF. DIF - Decentralized Identity Foundation, n.d. https://identity.foundation/.

[DIF_ION] Decentralized-Identity. Decentralized-Identity/Ion, n.d. https://github.com/decentralized-identity/ion.

[DIF_Sidetree] DIF. Decentralized-Identity/Sidetree, n.d. https://github.com/decentralized-identity/sidetree.

[DIF_Universal_Resolver] decentralized-identity. Decentralized-Identity/Universal-Resolver, n.d. https://github.com/decentralized-identity/universal-resolver.

[Dingle_2021] Dingle, Pamela. “ION – We Have Liftoff!” TECHCOMMUNITY.MICROSOFT.COM, March 2021. https://techcommunity.microsoft.com/t5/identity-standards-blog/ion-we-have-liftoff/ba-p/1441555.

[Finck_2019] Finck, Michèle. “Blockchain and the General Data Protection Regulation - Can Distributed Ledgers Be Squared with European Data Protection Law?,” July 2019. https://data.europa.eu/doi/10.2861/535.

[Frauenthaler_2019]Frauenthaler, Philipp & Sigwart, Marten & Borkowski, Michael & Hukkinen, Taneli & Schulte, Stefan. (2019). Towards Efficient Cross-Blockchain Token Transfers. 10.13140/RG.2.2.35827.27688.

[Frizzo-Barker_2019] Frizzo-Barker, Julie, Peter A. Chow-White, Philippa R. Adams, Jennifer Mentanko, Dung Ha, and Sandy Green. “Blockchain as a Disruptive Technology for Business: A Systematic Review.” International Journal of Information Management 51 (October 2019): 102029. https://doi.org/10.1016/j.ijinfomgt.2019.10.014.

[Giancaspro_2017] Giancaspro, Mark. "Is a 'Smart Contract' Really a Smart Idea? Insights from a Legal Perspective." Computer Law & Security Review 33, no. 6 (2017): 825–35. https://doi.org/10.1016/j.clsr.2017.05.007.

[Governance-Polkadot_wiki] Governance · Polkadot Wiki. · Polkadot Wiki, n.d. https://wiki.polkadot.network/docs/en/learn-governance.

[Govind_2020] Govind, Arjun "The Case for Self-Sovereign Identity." R3, April 2020. https://www.r3.com/blog/the-case-for-self-sovereign-identity/.

[Greenlight] GreenLight Dev Ledger Indy Network. GreenLight, n.d. http://dev.greenlight.bcovrin.vonx.io/.

[Guo_2019] Guo, Leiyong, Hui Xie, and Yu Li. "Data Encryption Based Blockchain and Privacy Preserving Mechanisms towards Big Data." Journal of Visual Communication and Image Representation 70 (December 2019): 102741. https://doi.org/10.1016/j.jvcir.2019.102741.

[Hoskinson_2020] Hoskinson, Charles. "Atala versus Cardano (Permissioned versus Permissionless)." YouTube, September 2020. https://www.youtube.com/watch?v=uZgDxPCXgPo.

[Hyperledger_AriesOpenAPIDemo] Hyperledger. Aries OpenAPI Demo, 2021. https://github.com/hyperledger/aries-cloudagent-python/blob/main/demo/AriesOpenAPIDemo.md.

[Hyperledger_ACA-Py] Hyperledger. Hyperledger/Aries-Cloudagent-Python, n.d. https://github.com/hyperledger/aries-cloudagent-python.

[Hyperledger_Aries-framework-dotnet] Hyperledger. Hyperledger/Aries-Framework-Dotnet, n.d. https://github.com/hyperledger/aries-framework-dotnet.

[Hyperledger_Aries-framework-go] Hyperledger. Hyperledger/Aries-Framework-Go, n.d. https://github.com/hyperledger/aries-framework-go.

[Hyperledger_Aries-RFCs] Hyperledger. Hyperledger/Aries-Rfcs, n.d. https://github.com/hyperledger/aries-rfcs.

[IOHK_AtalaPrism] IOHK. atalaprism.io. Atala Prism, n.d. https://www.atalaprism.io/

[Input-Output-HK] Input-Output-Hk. Input-Output-Hk/Plutus. GitHub, n.d. https://github.com/input-output-hk/plutus.

[IOHK_January2021] IOHK. Cardano Monthly Update - January 2021. YouTube, 2021. https://youtu.be/e_C90pUrfKs?t=4536.

[IOHK_March2021] IOHK. "Cardano360 - March 2021." Youtube, March 2021. https://www.youtube.com/watch?v=ULBLgPgxtN8.

[iohkdev.io] iohkdev.io. Alpha.Marlowe.Iohkdev.Io. Marlowe Playground, n.d. https://alpha.marlowe.iohkdev.io/.

[Jawaheri_2020] Jawaheri, Husam Al, Mashael Al Sabah, Yazan Boshmaf, and Aiman Erbad. "Deanonymizing Tor Hidden Service Users through Bitcoin Transactions Analysis." Computers & Security 89 (February 2020): 101684. https://doi.org/10.1016/j.cose.2019.101684.

[Jones_2019] Jones, Ben, Dan Robinson, Georgios Konstantopoulos, Joseph Poon, Karl Floersch, Kelvin Fichter, Vitalik Buterin, and Xuanji Li. Plasma Contracts. plasma.io, 2019. https://plasma.io/plasma-contracts.html.

[Karafiloski_2017] Karafiloski, Elena, and Anastas Mishev. “Blockchain Solutions for Big Data Challenges: A Literature Review.” IEEE EUROCON 2017 -17th International Conference on Smart Technologies, 2017, 763–68. https://doi.org/10.1109/eurocon.2017.8011213.

[Kiayias_2016] Kiayias, Aggelos, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol, 2016. https://eprint.iacr.org/2016/889.

[Lamport_2019] Lamport, Leslie, Robert Shostak, and Marshall Pease. “The Byzantine Generals Problem.” In Concurrency: The Works of Leslie Lamport, 203–26, 2019.

[Lee_2019] Lee, Jei Young. “A Decentralized Token Economy: How Blockchain and Cryptocurrency Can Revolutionize Business.” Business Horizons 62, no. 6 (November 2019): 773–84. https://doi.org/10.1016/j.bushor.2019.08.003.

[Li_2017] LI Yue, HUANG Junqin, Q. I. N. Shengzhi, and WANG Ruijin. “Big Data Model of Security Sharing Based on Blockchain.” 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), 2017. https://doi.org/10.1109/bigcom.2017.31.

[Liu_2019] LIU Zhiqiang, TANG Shuyang, Sherman S. M. Chow, LIU Zhen, and LONG Yu. “Fork-Free Hybrid Consensus with Flexible Proof-of-Activity.” Future Generation Computer Systems 96 (2019): 515–24. https://doi.org/10.1016/j.future.2019.02.059.

[Maller_2019] Maller, Mary, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. “Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings.” Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019. https://doi.org/10.1145/3319535.3339817.

[Mazières_2000] Mazières, David, and David Folkman. “Self-Certifying File System.” PhD Thesis, Massachusetts Institute of Technology, 2000.

[Mazières_1998] Mazières, David, and M. Frans Kaashoek. “Escaping the Evils of Centralized Control with Self-Certifying Pathnames.” Proceedings of the 8th ACM SIGOPS European Workshop on Support for Composing Distributed Applications - EW 8, 1998. https://doi.org/10.1145/319195.319213.

[Merkle_1990] Merkle, Ralph C. “A Certified Digital Signature.” Advances in Cryptology — CRYPTO’ 89 Proceedings, 1990, 218–38. https://doi.org/10.1007/0-387-34805-0_21.

[Microsoft_2019] Microsoft. “Decentralized Identity. Own and Control Your Identity,” 2020. https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE2DjfY.

[Nakamoto_2008] Nakamoto, Satoshi. “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. https://bitcoin.org/bitcoin.pdf.

[Polkadot_wiki] Polkadot Wiki · The Hub for Those Interested in Learning, Building, or Running a Node on Polkadot., n.d. https://wiki.polkadot.network/.

[Poon_2015] Poon, Joseph, and Thaddeus Dryja. “The Bitcoin Lightning Network.” Scalable O-Chain Instant Payments, 2015.

[Reed_2021] Reed, Drummond, Manu Sporny, Dave Longley, Christopher Allen, Ryan Grant, and Markus Sabadello. Decentralized Identifiers (DIDs) v1.0. W3C Working Draft 09 March 2021. W3C, 2021. https://www.w3.org/TR/2021/WD-did-core-20210309/.

[Sabadello_2021] Sabadello, Markus, and Dmitri Zagidulin. Decentralized Identifier Resolution (DID Resolution) v0.2. Credentials Community Group, 2021. https://w3c-ccg.github.io/did-resolution/.

[Simmons_2021] Simmons, Jake. "Demand for Cardano's Atala PRISM Outpaces Labor Supply." Crypto News Flash, April 2021. https://www.crypto-news-flash.com/demand-by-fortune-500-governments-for-cardanos-atala-prism-outpaces-labor-supply/.

[Sovrin_2020]Sovrin Foundation. Write to the Sovrin Public Ledger. Sovrin, 2020. https://sovrin.org/issue-credentials/.

[Sporny_2019] Sporny, Manu, Dave Longley, and David Chadwick. Verifiable Credentials Data Model 1.0. W3C Recommendation 19 November 2019. Verifiable Credentials Data Model 1.0. W3C, 2019. https://www.w3.org/TR/2019/REC-vc-data-model-20191119/.

[Steele_2020] Steele, Orie, and Manu Sporny. DID Specification Registries. W3C Editor's Draft 10 December 2020. W3C DID Specification Registries. W3C, 2020. https://w3c.github.io/did-spec-registries/.

[Stoll_2019] Stoll, Christian, Lena KlaaBen, and Ulrich Gallersdörfer. "The Carbon Footprint of Bitcoin." SSRN Electronic Journal, July 2019. https://doi.org/10.2139/ssrn.3335781.

[Sutherland_2019] Sutherland, Brandon R. "Blockchain's First Consensus Implementation Is Unsustainable." Joule 3, no. 4 (April 2019): 917–19. https://doi.org/10.1016/j.joule.2019.04.001.

[Swan_2015] Swan, Melanie. Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc., 2015.

[Szabo_1996] Szabo, Nick. Smart Contracts: Building Blocks for Digital Markets, 1996. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[Szabo_1997] Szabo, Nick. "The Idea of Smart Contracts." Nick Szabo's Papers and Concise Tutorials 6, no. 1 (1997).

[Tobin_2016] Tobin, Andrew, and Drummond Reed. "The Inevitable Rise of Self-Sovereign Identity." The Sovrin Foundation 29, no. 2016 (2016).

[ToIP_2020] ToIP. "Introducing the Trust over IP Foundation," May 2020.

[Viriyasitavat_2019] Viriyasitavat, Wattana, and Danupol Hoonsopon. "Blockchain Characteristics and Consensus in Modern Business Processes." Journal of Industrial Information Integration 13 (March 2019): 32–39. https://doi.org/10.1016/j.jii.2018.07.004.

[Wackerow_2021] Wackerow, Paul, Sam Richards, Stanislav Bezkorovainyi, Kevin Ziechmann, ethosdev, Nikolai Golub, Aheesh, et al. Layer 2 Scaling. ethereum.org, 2021. https://ethereum.org/en/developers/docs/layer-2-scaling/.

[Wolfson_2019] Wolfson, Rachel. "Cardano Founder Launches Enterprise Blockchain Framework In Collaboration With Ethiopian Government." Forbes. Forbes Magazine, April 2019. https://www.forbes.com/sites/rachelwolfson/2019/04/30/cardano-founder-launches-enterprise-blockchain-framework-in-collaboration-with-ethiopian-government/?sh=4683cdaf4e10.

[Wood_2014] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151, no. 2014 (2014): 1-32.

[Zachariadis_2019] Zachariadis, Markos, Garrick Hileman, and Susan V Scott. "Governance and Control in Distributed Ledgers: Understanding the Challenges Facing Blockchain Technology in Financial Services." Information and Organization 29, no. 2 (June 2019): 105–17. https://doi.org/10.1016/j.infoandorg.2019.03.001.

[Zhang_2019] Zhang, Bingsheng, Roman Oliynykov, and Hamed Balogun. "A Treasury System for Cryptocurrencies: Enabling Better Collaborative Intelligence." In The Network and Distributed System Security Symposium 2019, 2019. https://doi.org/10.14722/ndss.2019.23024.

[Zheng_2017] Zheng, Zibin, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends." In 2017 IEEE International Congress on Big Data (BigData Congress), 557–64. IEEE, 2017. https://doi.org/10.1109/bigdatacongress.2017.85.

[Zilliqa_2017] Zilliqa, Team and others. "The Zilliqa Technical Whitepaper." 2019.

Appendix A - Zero-Knowledge Proofs and zk-SNARKs

Researchers at MIT in 1985 first proposed Zero-knowledge proofs (ZKPs). Initially, ZKPs were designed for interactive proof systems, having a Prover and a Verifier. The goal was for the Prover to prove knowledge of information x to the Verifier without communicating any information other than the fact that 'Prover knows x.'

By definition, a ZKP must satisfy:

- **Completeness:** If Prover genuinely holds the knowledge and both parties follow the protocol correctly, Verifier can be eventually convinced without any external help.
- **Soundness:** Prover can only convince Verifier if they hold the knowledge.
- **Zero-knowledge:** Verifier does not learn anything about the knowledge, simply that the Prover has said knowledge.

As an illustration, Prover is given by Verifier a graph with nodes and edges and is tasked to find a 3-coloring such that no two adjacent nodes have the same color. Imagine Verifier is an organization, and Prover is a freelancer. Prover finds a 3-coloring and must prove that they have done so without showing the result before getting paid.

To construct an interactive ZKP protocol, the graph is drawn on the floor in a closed room. Prover, alone, places colored balls on the nodes according to the discovered pattern and covers the balls with non-transparent bowls. Verifier comes into the room, chooses an edge, and verifies that the attached nodes are different colors. If they are not different, then Prover is disqualified. See Figure A.1.

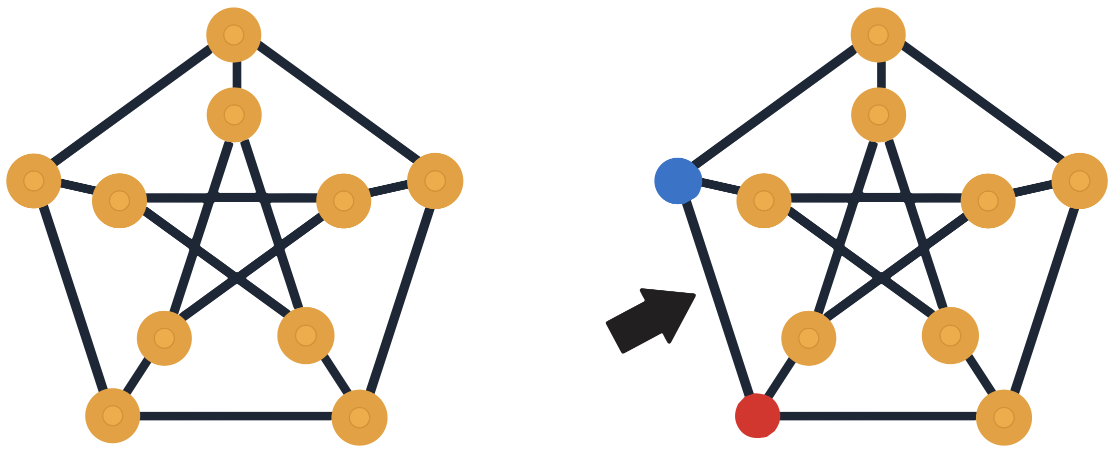


Figure A1: Interactive ZKP Illustration

Since a single attempt by the verifier, checking adjacent nodes of one edge is not probabilistically certain, the verifier must be allowed as many checks till satisfaction. However, if the verifier were to record the colors and check all edges secretly, they would consequently have violated the zero-knowledge principle. To rectify this, the verifier must leave the room after validating an edge. The prover may then switch the colors of the balls, maintaining the pattern. Thus, proving that all edges chosen consistently have two different colored edges but maintaining zero knowledge of the exact pattern. This can be done until the verifier has certainty that there is only negligible error.

The problem with traditional ZKP is that it is an interactive system, requiring Prover and Verifier's presence, and as such is not scalable. Zk-SNARKs or zero-knowledge Succinct Non-interactive ARguments of Knowledge provides an algorithmic formation using cryptographic hash functions.

The three algorithms in zk-SNARK:

- Prover (P): takes in three inputs – proving key (pk), a random input (x), and the statement to prove (s). Such that P(pk, x, s) = prf.
- Verifier (V): takes in the random input (x), verifying key (vk), and prf from the P algorithm and returns a Boolean answer according to validity.
- Key Generator (G): takes in a secret value (λ) and a program (C) to generate pk and vk. The lambda value has to be kept private and confidential as any holder could generate proving and verifying keys and build counterfeit proofs.

The following steps utilize the above algorithms:

- The verifier puts in a secret λ value to generate the proving and verifying keys: G(C, λ) = (pk, vk).
- The prover must take the pk and prove their statement: prf = P(pk, x, s). Here x is the SHA-256 hash of s.
- When prf is sent to the verifier and inserted into the verifying algorithm: V(vk, x, prf) = True if the prover is honest, False otherwise.

Appendix B - GDPR

The EU's 2016 General Data Protection Regulation (GDPR) is the forerunner of human-centric data protection regulations.

While the GDPR was written using technology-agnostic wording, there are two main points of tension between its regulations and DLT:

- The GDPR's assumption that there is at least one natural or legal person – data controller – whom a data subject can address to enforce their rights over every piece of personal data.
- The GDPR requires that data can be modified or erased.

GDPR-based rights of a data subject:

- The right to access (Article 1)
- The right to rectification (Article 16)
- The right to erasure (Article 17)
- The right to restriction of processing (Article 18)
- The right to portability (Article 20)
- The right to object (Article 21)
- The right to not be subject to a decision based solely on automated processing, including profiling, which significantly affects them (Article 22)

Blockchains as a whole are not either compliant nor incompliant with the GDPR [Finck_2019] since SSI using blockchain technology can be seen as attempting similar purposes as the GDPR. However, further legal cases or further addressing need to be done to prevent the EU from falling behind in technology development.

Issues that need clarification include the following [Finck_2019]:

- Is anonymization an effective means of provoking the 'erasure' of data for the purposes of Article 17 GDPR?
- Can a peppered hash produce anonymous data?
- What is the status of the on-chain hash where transactional data is stored off-chain and subsequently erased?
- What is the status of anonymity solutions such as zero-knowledge proofs under the GDPR?
- Can a data subject be a data controller in relation to personal data that relates to themselves?
- Is the off-chain storage of transactional data a means of complying with the data-minimization principle?
- How ought 'erasure' to be interpreted for the purposes of Article 17 GDPR? Can the deletion of a private key satisfy as the erasure of on-chain data?

Appendix C - Recovery Key and Backup Management

Possible places for wallet backup storage:

1. The device itself
2. The cloud agent
3. Another device
4. Peer-to-peer files

The device itself is an easy location to store the encrypted backup. The device agent would initiate an incremental backup of the KMS database. This has an advantage for easy restoration should there be a problem with the wallet app and prepackaging in advance of external (cloud agent) backup. The disadvantage is primarily if the device is lost or irreparably damaged.

In the case of a mobile provider, the cloud agent would be an obvious place to store an encrypted backup. The potential risk lies in the cloud agent service operator having access to the data, even in encrypted form, as they could work on cracking the encryption. Alternatively, a separate trusted service such as a bank could potentially offer such services.

Another device or other hardware that you own or one controlled by a trusted other could be a valid backup. This could provide a recourse should your primary device be lost or broken.

Parts of the backup could be redundantly distributed to different backup holders via a P2P sharing technique, requiring collecting a certain number of the redundant files to restore the backup on the target device.

Recovery key management is as essential as a reliable backup location. Locations to store include:

- Hot storage – on device: protected via hardware provided protection like biometrics. This would prove moot should the device be lost.
- Cold storage – on paper or USB flashdisk: stored to a device that is not online, eliminating remote access by an attacker. On paper could be risky as, over time, the ink may fade. Storage on a USB key could be sufficient, though finding a safe place to store it may be an issue in itself. A bank deposit box or keeping a sealed copy with a lawyer is a possibility that may be costly. Keeping a copy at work or with a friend/family member or two is another alternative. Though the more copies around, the higher the risk of unauthorized use.
- Sharded hot storage – social recovery: a novel proposal to shard the key into pieces and distribute the pieces amongst a set of holders, requiring only a decided subset of the pieces to combine for full key restoration - see passguardian.com.
- Memorizing: this may be difficult to retain as the key must be sufficiently long and random to be effective.

Should all backups and recovery techniques fail or not exist, the wallet would be gone. In that case, a new wallet must be created and all credentials recollected from the various issuers. Presumedly the first credential would be a governmental ID and a degree of hassle and time to get the others in order of importance. Thus, multiple safe backups are necessary.

Appendix D - InterPlanetary File System

HTTP is the protocol behind client-server architecture that we see in the internet of today. However, it has many drawbacks; in particular, the centralized nature of all information that we access to over the web is held on servers that are under control of a central company. This requires trust that the central company will not go offline either through an attack or simple server failure. It also enables censorship, as the data stored on a central server can be easily blocked by ISPs blocking the server's IP. As all data is stored on the server, if the server goes down that data is no longer accessible.

Inter-Planetary File System (IPFS) introduces a peer-to-peer (P2P) distributed file system, creating a system by combining a P2P BitTorrent swarm with a Git repository. IPFS forms a Merkle DAG data structure upon which one can build distributed versioned file systems, with no need for trust between nodes, thereby escaping a single point of failure(SPF). IPFS aims to replace HTTP by offering an enhanced web combining successful file distribution techniques invented in the past fifteen years [Benet_2014].

A summary of the successful techniques that IPFS incorporates:

Block Exchanges – BitTorrent

BitTorrent [Cohen_2003] is a P2P filesharing system that coordinates untrusted peers (swarms) to distribute pieces of files collaboratively. Features from the BitTorrent ecosystem that IPFS assimilates include:

- BitTorrent's data exchange protocol that rewards nodes who contribute to each other and punishes nodes who only leach.
- Bit Torrent peers track availability of file pieces and prioritize the rarest pieces to be downloaded first. This allows leaching peers (those that do not have a full copy) to be capable of trading with each other.

Distributed Hash Tables

Distributed Hash Tables (DHT) regulate and maintain metadata for P2P filesystems. BitTorrent uses MainlineDHT to track peers that are part of a torrent swarm.

Version Control Systems – Git

Version Control Systems model files changing over time and distribute all versions accurately. Git provides a Merkle DAG [Becker_2008] object model that captures changes to a file system tree. Files are objects that are content-addressed via a cryptographic hash of their contents. Links to other objects are embedded in a Merkle DAG. Versioning metadata, including branches and tags, are pointer references which are low in memory and thus easy to create and update. Version changes update references or add objects. Distributing version changes is a simple transfer of objects and update of remote references.

Self-Certified Filesystems – SFS

SFS proposed a global file system separated from key management. In SFS, file names contain the public keys thereby making them self-certifying pathnames. Key management occurs in user generated procedures outside the SFS file system, enabling a versatile file system with interchangeable key management mechanisms [Mazières_1998, Mazières_2000]. SFS allows distributed systems outside of centralized control [Merkle_1990].

SFS introduced the following remote filesystem-addressing scheme:

```
/sfs/<Location>:<HostID>
```

where Location is the server network address and HostID = hash(PublicKey, Location).

The name of an SFS file system certifies its server by offering the server's public key that can be verified by the user. SFS instances have cryptographic namespace rather than one controlled by centralized authorities.

Content-Based addressing vs Location-Based addressing

In the web of today, individuals browse to a Location (e.g. https://www.someserver.com/somefile.html) where they can access the information that is stored on the server, and their browser then stores a copy in their cache. Unless the Location points to a static file that some user has recently stored in their uncleared cache, that information will be unobtainable should the server go down.

More significantly, if one user had a cached copy of a static file they would be unable to share it with another user without retrieving the static copy from their cache and hosting the copy on another server (dismissing copyright infringement). In the event that a user would re-host the file on a different server, the Location would be different, and thus would be inaccessible to any user who only had the Location of the unobtainable server.

IPFS introduces Content-Based Addressing (CBA), also know as Name Driven Networking (NDN). With CBA when requesting a specific resource, the exact file by its hash (aka. fingerprint) is what is needed rather than the Location. When the IPFS network is asked for a file by a requester and someone on the IPFS network provides the resource, the resource will be copied to requester's cache allowing the requester to provide a future requester the same content. This system will increase in speed should the file be shared amongst a larger quantity of nodes.

The security ensuring that a file has not been tampered is the hash, or content name. When asking for a specific hash name, the file received can be checked to confirm that it has the same hash. If the hashes are equivalent, the file has not been altered.

IPFS also provides deduplication. If multiple users post the same file, it is stored only once in the IPFS network as it has the same hash. This improves efficiency across the network.

IPFS Design

The IPFS protocol is a stack of sub-protocols that handle functionality [Benet_2014]:

1. **Identities** – node identity formation and validation
2. **Network** – administers connections between peers
3. **Routing** – manages information to locate peers and objects
4. **Exchange** – a block-exchange protocol (BitSwap) that conducts productive block distribution
5. **Objects** – a Merkle DAG of links to content-addressed immutable objects
6. **Files** – Git-like versioned file system
7. **Naming** – self-certifying mutable name system

Identities

A cryptographic hash of a public key, NodeId, identifies nodes. Although users are able to reinitialize a new identity on each launch, they are incentivized to retain the same identity due to loss of accrued network benefits. On first connection, peers exchange public keys and check if the hash other.PublicKey is equal to other.NodeID.

Hash values are stored in multihash format:

```
<function code><digest length><digest bytes>
```

Multihash enables system flexibility to change function depending on the use case and allow for future upgrades.

Network

IPFS connects hundreds of nodes in the network or across the internet.

Including features:

- **Transport:** using any transport protocol
- **Reliability:** using UTP if underlying networks do not supply another
- **Connectivity:** using ICE NAT traversal techniques
- **Integrity:** optionally checks messages against a hash checksum
- **Authenticity:** optionally digitally signs messages with sender's private key

Routing

In order to provide nodes with connections to other peers' objects and determine which peers can serve desired objects, IPFS uses a DSHT based on a combination of S/Kademlia and Coral. Values <= 1KB are stored directly on the DHT, and for larger values a reference (NodeIds of peers who can serve the block) is stored on the DHT.

IPFS uses BitSwap, based on BitTorrent, as a transport protocol for exchange. BitSwap peers have two sets of blocks ({want_list}, {have_list}). Contrary to BitTorrent, BitSwap is not limited to blocks in the same torrent. BitSwap operates as a marketplace for all blocks where nodes can acquire necessary blocks. Nodes barter in the marketplace for block exchange. See Filecoin [Filecoin] for current development. In basic use, peers must have blocks that its peers want in order for complementary exchange. This aims to incentivize nodes to cache rare pieces to use for exchange even if they have no need for the pieces directly.

Objects

IPFS builds a Merkle DAG with links between objects as cryptographic hashes. This provides:

- Content addressing, with all content uniquely identified by its hash checksum.
- Tamper resistance, with all content verified by its checksum and invalid or tampered data is rejected.
- Deduplication, with all objects that have the same checksum stored only once.

Each Object stores 256kb of data. An object can also contain a link to another IPFS Object, making it possible to store data that is larger than 265kb. A larger file would be broken into as many Objects necessary to hold the file size, and an Empty object linking to all the Objects that the file was broken into.

Files

IPFS defines a versioned filesystem on top of the Merkle DAG similar to Git:

- **File Object:** blob – contains an addressable unit of data representing a file.
- **File Object:** list – represents a collection made up of IPFS blobs or list objects concatenated together. An IPFS list functions like a filesystem file with indirect blocks. Directed graphs where the same node appear in multiple places allows in-file deduplication.
- **File Object:** tree – represents a directory or map of names to hashes that represent blobs, lists, other trees, or commits.
- **File Object:** commit – represents a snapshot in the version history of an object.

Naming

Inter Planetary Naming System (IPNS) tracks immutable IPFS objects with a mutable global namespace, following the naming scheme from SFS.

In IPFS, NodeId = hash(node.PubKey). Every user is assigned a mutable namespace (`/ipns/<NodeId>`) which enables them to publish an object signed by user's private key. Other users can verify the object and confirm that it matches the public key and NodeId to verify the authenticity of mutable state retrieval.

The separate prefix allows distinction between mutable (`ipns`) and immutable (`ipfs`) paths. The process follows publishing the object as an immutable IPFS object and then publishing its hash on the routing system as a metadata value. An optional commit object to store a version history is advised when necessary.