Findora Litepaper

Findora Foundation

Version 3.2 (November 2020)

Abstract

Findora is a fully confidential yet auditable, high-throughput, and scalable public financial infrastructure. The *cryptographically transparent* financial ledger at Findora's core enables efficient, accessible, and transparent financial services.

Blockchains have drawn attention for their far-reaching financial applications, stemming from their capability to improve transparency, trust, and coordination in transactional systems. Findora targets sectors of financial infrastructure that most desperately need improved transparency, but whose privacy and compliance requirements are not met by existing public blockchains. These include financial services such as investment funds, lending platforms, and security exchanges that opaquely handle trillions of dollars worth of assets, are highly susceptible to fraud, and often very inefficient.

Findora envisions a world where financial networks are compliant and publicly auditable at all times. Packaged with every asset are rules governing ownership, transferability, and compliance. A network of validators enforces these rules while distributing trust. Imagine a lending platform proving solvency, a fund proving it is investing within its mandate, an investor anonymously demonstrating accreditation, or a regulator using fine-grained auditing keys, all while completely confidential.

Contents

1 Blockchains and finance

Blockchains are a technology that promises to improve transparency and efficiency in financial systems. Bitcoin, the pioneering cryptocurrency, arose out of an initiative to create a new global currency free from the governance or control of any select privileged organization. As the technology advanced to include many new capabilities including smart contracts, zero-knowledge transactions, and multi-signature wallets, the industry came to realize far-reaching potential applications beyond decentralized currencies.

1.1 Protocol standards

In many ways, financial infrastructure can be compared to the Internet in the days before widespread adoption of standard protocols such as TCP/IP, TLS, HTTP, and SMTP. The most basic requirement for a network to function without centralized coordination is the adoption of a common language, so that independent systems can talk seamlessly to each other and share information. For email, SMTP provided a standard data transfer protocol so that messages could be sent between different mail clients and servers. In financial systems, there is the additional constraint of interdependent information transfer. For instance, a double bank transfer of the same electronic funds from one bank to two different banks would be problematic.

Full coordination between a set of financial institutions entails synchronizing their local databases. The SWIFT network and standardized codes were developed in the 1970s to support interbank communication, but functions only as a specialized messaging service for banks. Although distributed consensus protocols like Paxos have existed for decades, it was not until recently with the advent of blockchain consensus that new protocol standards for atomic asset transfers and consensus were developed.

The recent explosion of the blockchain and cryptocurrency industry has generated the opportunity for a long-awaited makeover of the worldwide financial system. In a short period of time, it has spurred the development of new open source infrastructure for distributed databases, peer-to-peer broadcast protocols, and Byzantine consensus algorithms. New library implementations of cryptographic tools have also emerged, such as multi-signatures and zero-knowledge proofs, improving digital integrity and privacy.

1.2 Financial identity

Developing one's financial identity on the Internet presents unique challenges around transferability, security, and privacy. Identities are central to the Internet. Each Internet-connected device has an IP address and each Internet user establishes a number of identities online. Depending on the type of online platform, a user's identity can contain varying degrees of personally identifiable information or attributes that will be manually or automatically captured each time he or she joins a new platform. Moreover, ones's online identity cannot be transferred easily to another platform. As with most data on the Internet, financial and identity data lives in data siloes owned by institutions and service providers rather than users themselves.

In the context of financial systems, identity has been traditionally necessary for ownership of assets. Apart from physical assets, establishing ownership relies on legal documents that reference real-world identities. Cryptocurrencies have pushed the adoption of public-key cryptography as an alternative method of establishing ownership. Every Bitcoin is owned by a unique public key. Each public key has a corresponding private key that can be used to create a *digital signature* on any statement. A digital signature is unforgeable without knowledge of the private key, but can be verified given only the public key. Bitcoin transactions that transfer ownership of a Bitcoin from one public key to another require a valid digital signature using the private key corresponding to the first public key. This method of establishing ownership may be equally applied to more diverse assets. Company stock can be legally issued to a public key so that only someone who knows the corresponding secret key may claim ownership.

However, ownership of diverse assets in a global financial system typically has stricter identity and account validation requirements compared to other domains. For instance, banks and brokerage services are required to comply with know your customer (KYC) and anti-money laundering (AML) rules before accepting new clients. One's credit history is tied to his or her social security number and one's accreditation status is tied to his or her annual income or net worth.

The blockchains behind cryptocurrencies do not address the identities of their participants in this detail. Public keys alone do not contain any information about a user. Importantly, a user can easily add additional information about their identity to a public key, which they can provide access to. This identity can enable the enforcement of a rich set of rules surrounding KYC, AML, and other accreditation that gate who can participate in

different types of financial transactions. However, no existing blockchain has proposed a comprehensive standard for real world identity management and attestation.

Public-key cryptography also allows individuals to truly own and manage their own identity. The pervasive modus operandi today is for data providers to manage identity on behalf of users, leaving users with no control. Such systems often profit when users are prevented from carrying their identity elsewhere. A Facebook identity is isolated from a Twitter identity. A customer of JPMorgan Chase cannot easily port their data and identity to Bank of America. The popular framework OAuth is used for cross platform identity attestation, but involves the identity manager as an intermediary. In contrast, public-key cryptography enables users to obtain digitally signed attestations from identity providers and manage these on their own. It is nonsensical that a user who is able to access their Wells Fargo bank account balance online every day over an authenticated Internet channel cannot easily share this data with someone else without the involvement of intermediaries. This would be solved by a simple digital signature from the Wells Fargo web server on a timestamped account balance.

1.3 Transparency and synchrony

Whether managed by a single server, a consortium, or a massively distributed network of operators, all blockchain networks have one thing in common: the *blockchain* data structure. A blockchain is a simple authenticated data structure: an append-only list of records cryptographically linked together that facilitates consensus on the history of transactions in a shared open database. The *head* of the blockchain is a single 32-byte number that summarizes the entire list of records. It is generated using a collision-resistant hash function and it is virtually impossible to create two different lists of records that give the same *head* number. Verifying that operators in the network agree on the same list of records only requires comparing this one number. Moreover, anyone who knows the "correct" head of the blockchain can download the list of records from an untrusted source and verify its correctness. The head is updated each time a new record is appended to the list. The updated head can be computed without knowledge of the entire list of records. Once a node is synchronized with the operators of the blockchain, it can follow along as new records are appended.

The blockchain data structure allows anyone to easily synchronize with the blockchain operators and maintain an authentic local copy of the database.

However, it does not immediately enable someone who only knows the head of the blockchain to retrieve only a portion of the database (e.g. the balance in an account) and verify that the content received is authentic (i.e. consistent with the head). Most users will not store a synchronized local copy of the database and therefore fall into this category. A more advanced authenticated data structure, such as a Merkle tree, can solve this. A Merkle tree summarizes the entire *state* of the database (e.g. the balance in all accounts) in a single 32-byte number called the *state commitment*. There is a procedure to generate a proof that any database lookup is correct, which can be verified against the commitment. The proof is relatively small (i.e. less than 2 KB).

A data provider that stores the entire database may answer user queries to the database with these authentication proofs. Users can verify the authenticated query against the state commitment, which shows that the query response is correct and consistent with the same database replicated on all other nodes in the network. As with the blockchain head, the state commitment is updated with each new block of transactions. Users who submit transactions may also obtain a proof that the updates were incorporated into the new state commitment.

In summary, authenticated data structures make it easy for a network of operators, users, and auditors to have open access to a common financial database without trusting a central server to provide access to the data honestly. The history of transactions is immutable and anyone can verify that all transactions are valid. Findora's advanced authenticated data structures are based on state-of-the-art techniques from cryptographic accumulators and vector commitments.

1.4 Privacy and compliance

Transparency and open-participation are cornerstones of blockchain-based finance. However, full transparency comes at the cost of privacy. Existing systems that provide a barebones infrastructure for blockchain-based financial services include Ethereum, Ripple, Hyperledger Fabric, and Stellar, but none of these systems provide confidentiality. Privacy is an absolute requirement for the vast majority of financial services, and thus despite being auditable, these platforms are immediately disqualified for many use cases. Cryptocurrencies such as Zcash and Monero utilize cryptography to preserve the confidentiality and anonymity of coin transfers. However, these systems are limited in that confidentiality is all-or-nothing; transactions only prove that a simple transfer of coins is valid, but cannot prove more nuanced state-

ments. Additionally, Monero and Zcash only natively support their native cryptocurrencies.

Traditional finance gives users privacy from the public, but offers close to zero transparency and does not protect user privacy from institutions. First generation blockchain-based finance offered either full transparency or complete confidentiality. Findora aims to provide the best of all worlds with what we call *cryptographic transparency*.

Cryptographic transparency offers better privacy than traditional finance, protecting detailed user data even from the operators of the infrastructure, while still allowing operators to verify the validity of all transactions. It maintains the transparency and auditability of first generation blockchains, enabling users to prove complex statements about the details of private transactions. Findora provides privacy-centric tools for asset tokens, identity/KYC integration, public and regulatory audits, asset tracking, and many other special purpose zero-knowledge proof capabilities for demonstrating that transactions are compliant: for example, proving an exchange is solvent or that a fund is invested properly in whitelisted assets.

1.5 Consensus

One of the greatest misconceptions regarding blockchains is that they can simply replace trust in financial institutions. In fact, the network that operates a blockchain is itself the new financial institution. The consensus protocol on which it operates determines how influence is distributed among the network participants.

1.5.1 The case for distributed consensus

Consensus protocols are manifest in organizational design. Even centrally controlled organizations feature elements of consensus. The processes for electing officials in a national government or the members of a company board and deciding on national policies or company direction are all forms of consensus protocols. An autocracy has no consensus protocol. An autocratic leadership is overthrown only through revolution, which is an inherently chaotic form of change. Revolutions have high social costs, often inhibiting their formation. A consensus protocol is an algorithmic way to make organizations more elastic and reduce the cost of change.

Decentralization Consensus protocols vary greatly depending on the organizational goals as well as participation. *Decentralization* broadly describes

a consensus participation that is more open and distributed. Pure democracies are more decentralized than republics. Cooperatives (e.g. credit unions) are more decentralized than privately-help corportations. The minting of new Bitcoins is arguably more decentralized than the printing of US dollars. Anyone with access to electricity can participate in Bitcoin mining. On the other hand, Bitcoin minting in practice is dominated by mining rigs in China and highly influenced by developer protocol updates[1]. There is no uniform definition of decentralization.

Risks of centralized control When a financial ledger is operated by a single privately-held company, this company has full control over operating the ledger correctly and fairly, including setting transaction prices. If the company sets prices too high, censors transactions, or is caught accepting invalid transactions, then the only recourse users have is to join a new ledger system operated by a competitor. Similar to revolution, this may come at prohibitively high cost to the user. If PayPal suddenly raised transaction fees and put restrictive limits on account withdrawals, users may be simply unable to transfer their assets cheaply to their banks.

While something this dramatic may seem unlikely to happen to US domestic users, it is a realistic concern with cross border services. Recently, the popular Binance cryptoasset exchange announced a 90-day grace period until its US users will no longer be able to trade assets in their accounts or withdraw beyond limits.[2] Furthermore, financial ledgers have so-called "network effects", just like social networks, where the value of using the network increases with the number of users. Once dominant, network effects allow monopolies to thrive in spite of popular disapproval (e.g. the 2017 Equifax data breach[3], Facebook's data sharing revelations[4]).

Analogy to co-ops Findora's vision for a public financial infrastructure is akin to a large financial cooperative with worldwide participation, owned and democratically controlled by its users and operatives. A financial ledger controlled by a cooperative is already more immune to the risks of privately-managed centralized ledgers.

[1] https://en.wikipedia.org/wiki/SegWit

[2] https://www.coindesk.com/binance-exchange-to-block-crypto-trading-for-u-s-customers

[3] https://www.bloomberg.com/news/articles/2018-09-07/equifax-breach-a-year-later-record-profits-share-price-revival

[4] https://www.nytimes.com/2018/12/19/technology/facebook-data-sharing.html

A cooperative is run democratically and can vote to replace deviant operators. One might have a board elected by its members, which appoints an operator to run the ledger system. If the operator misbehaves the board can remove the operator and appoint a new one. If the board is lax to replace an unpopular operator, its members may re-elect the board. While this might work in an idealized setting, it presupposes an unspecified mechanism to facilitate the member voting and operator replacement processes. In practice, without a trusted facilitator the process of replacing operators would likely be slow and disorganized. The ledger system would be unstable while voting to replace operators. Furthermore, running a fair election in a truly open-membership cooperative is notoriously difficult due to "Sybil attacks". A single individual could join the cooperative under many different member identities in order to amass more votes. This would be difficult to prevent at global scale. A well-designed operation-level consensus protocol helps solve these problems without relying on a trusted facilitator.

Operation-level consensus protocol The purpose of operational-level consensus is to integrate voting/re-election more seamlessly with day-to-day operation. In most blockchain ledger systems this is feasible because operators play a simple and objective role. Operators run automated transaction validation software and append any valid transactions to the ledger on a first-come-first-serve basis. Furthermore, operators are replaceable: anyone should be able to play the role of an operator without special permission. This stands in contrast to a traditional financial infrastructure, where operators must be trusted institutions and authorized to handle private information. In a blockchain ledger there is no differentiation between the information an operator can see versus the public. Even in a private ledger, such as Findora's, operators are able to verify transaction validity without seeing the transaction's private contents, using the magic of cryptography.

The operational-level consensus protocol should provide an efficient mechanism for all members of the cooperative to vote on each batch of transactions appended to the ledger. It introduces "competition" among operators that favors low prices and correct, objective behavior. However, there are difficult choices to be made when selecting a consensus protocol. The following sections will survey the diversity of options in state-of-the-art consensus protocols.

1.5.2 Consensus protocol participation

Classical consensus protocols were designed for a fixed set of voters who collectively propose, ratify, and ultimately reach agreement on statements. This is sometimes called *permissioned consensus*. We will call these voters *validators* because in our context they will validate blockchain transactions. Byzantine fault tolerant (BFT) consensus protocols are robust to *Byzantine faults*, where a portion of the validators might not only fail but also become corrupt and attempt to either subvert consensus or accept invalid/conflicting transactions. The non-corrupt validators are called *honest* validators. BFT consensus protocols enable honest validators to reach agreement and make progress even in presence of Byzantine faults; however, the precise conditions under which these protocols can succeed have been the subject of an ever-evolving body of research. Permissioned BFT protocols, such as PBFT[5], guarantee agreement among honest validators so long as fewer than 1/3 experience Byzantine faults and make progress under good network connectivity conditions.

Permissioned BFT is appropriate for a blockchain operated by a consortium of predetermined vaidators. In this setting, security is simple to reason about. It is based on distributed trust, and doesn't involve any complicated incentive compatibility arguments. The system is secure if the consortium is chosen well such that less than 1/3 of the validators would maliciously collude at any point. However, permissioned consensus has its limitations. First, it presupposes a validator selection process that may be unrealistic, especially for a global blockchain. How can the validators be chosen such that every user around the world trusts at least a 2/3 fraction? Furthermore, this selection process does not support the vision of a public financial infrastructure in which anyone can participate.

Nowadays, we have consensus protocols for more flexible validator sets. Often referred to as "unpermissioned" consensus protocols, these protocols change the permissioning model in a variety of incomparable ways. One generalization called *federated Byzantine agreement* (FBA) allows each validator to choose its own trust circle. Specifically, a validator may define for itself what constitutes a *quorum*, i.e. a set of nodes it would trust and agree with. In classical BFT protocol any 2/3 set of the validators is a quorum. In an FBAS system, one validator may believe that a quorum is 1/2 of a US bank consortium while another validator believes a quorum is 2/3 of an international bank consortium. Consensus works if validators make wise trust

[5]Castro, Miguel, and Barbara Liskov. *Practical Byzantine fault tolerance.* OSDI 1999.

choices and there is sufficient intersection among trust circles.[6] This type of consensus is used by the Stellar network.

Proof-of-work blockchain networks like Bitcoin and Ethereum distribute influence over a decentralized, anonymous body of "miners" who contribute computational resources to the network and are rewarded for reaching consensus. Proof-of-space blockchain networks such as Chia, Filecoin, or Spacemesh will rely on similar security assumptions, replacing computational power with data storage resources. Proof-of-stake blockchain networks such as Peercoin, and several other under development (e.g. Cardano Shelley, Ethereum Casper, Algorand, Dfinity) directly place influence in the hands of the stakeholders of the network's native digital currency. Specifically, voting power is allocated proportional to ownership of the native asset, and in some cases restricted to illiquid accounts "staked" as collateral.

1.5.3 Incentive compatibility

The security of Bitcoin or Ethereum relies either on the assumption that the majority of the mining power is controlled by non-malicious actors, or at least that the majority of miners are rational actors who are incentivized by mining rewards to maintain the integrity of the network. In proof-of-stake consensus, it is reasoned that the network's stakeholders have the strongest incentive to maintain the network's integrity and value.

The incentive compatibility underpinning the security of these systems, which distribute voting power according to proof-of-work or proof-of-stake, would be challenged if the potential gain from a malicious action were to exceed the value of the network itself. The miners' rewards or stake could lose their relative significance. This is particularly a concern if the consensus network is used for managing assets beyond the native currency, which derive their value externally. For this reason, pure incentive-backed consensus networks appear a risky choice for managing the $100+ trillion assets held in securities and traded in the world's capital markets.

On the other hand, it is important to recognize that a blockchain for recording ownership of external assets has a fundamentally different significance than a blockchain used only to record a native asset. The Bitcoin blockchain is the only source of truth for 'Bitcoin' ownership. A fork in the Bitcoin network results in two new independent versions of the Bitcoin asset

[6] David Mazières. *The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus.* 2014.

(e.g. Bitcoin and Bitcoin Cash) that are each less valuable than the original. In contrast, assets such as real-estate, company stock, or fractions of a dollar-backed reserve depend on external entities for enforcement, such as jurisdictional courts. A blockchain provides compelling *evidence* of ownership insofar as the relevant counterparties and enforcement agencies trust in, or in fact verify, its integrity. A perfectly honest blockchain provides strong evidence. A corrupt blockchain that accepted double-spends or otherwise blatantly invalid transactions does not. From this perspective, a malicious operator will more likely suffer from the system's loss of integrity than successfully steal a house over the blockchain.

To address the unique challenges in blockchain consensus for real-world assets, Findora's consensus protocol *Finsense* will integrate elements from both proof-of-stake consensus and FBA networks, where reputation and trust relationships play a greater role.

1.5.4 State machine replication

We will focus on BFT consensus for a transactional ledger, also known as *state machine replication*. The transactional ledger consists of a linear chain of transaction blocks that have been confirmed. Validators receive transactions, propose a block of transactions to append to the linear chain, and broad cast these proposals to other validators. Validators *confirm* a new block when they believe all other validators have agreed on the same block proposal. The consensus protocol should ideally allow validators to agree on the current state of the ledger, or equivalently the chain of historical confirmed transactions, as well eventually confirm new blocks within a reasonable amount of time.

Validators are either *honest* or *corrupt*. Honest validators are those who follow the protocol honestly and corrupt validators may exhibit arbitrary behavior. Honest validators might still experience network faults, such as going offline, or even become partially disconnected from only a subset of the other honest nodes in the network. The network is called *strongly synchronous* when all nodes are able to send messages to one another that are received within a known network time delay. The two basic goals of BFT consensus are *consistency* and *liveness*. Consistency is the property that all honest validators agree on the state of the ledger. Liveness is the property that the system doesn't get stuck, i.e. honest nodes are able to make progress and confirm new transaction blocks within a reasonable amount of time. Combining these two properties, if a new block is confirmed by one honest validator, it should eventually be confirmed by all the other honest validators.

Not surprisingly, no protocol can guarantee both consistency and liveness under all network conditions. If there is a partition among network validators (e.g. validators in China become disconnected from validators in the US), then each group of validators must decide between stalling indefinitely or continuing to confirm transactions at risk of losing consistency. This is one reason there is a diversity of consensus protocols, which all achieve consistency and liveness under different assumptions about the underlying network. Even protocols designed to resume liveness whenever the network regains full connectivity cannot guarantee consistency if 1/3 or more validators are corrupt.[7] In a protocol with a stake-based or work-based permissioning model, 1/3 corruption means that corrupt validators control 1/3 of the total stake or work. However, assuming the protocol is able to predict and hard code the maximum length of any network disruption, then it is possible to retain consistency among honest validators no matter how many other validators are corrupt.[8]

1.5.5 Properties of consensus protocols

Efficiency of massive consensus Counting votes among a potentially enormous number of validators would be impractical. Consensus protocols over arbitrarily large validator sets sometimes work by randomly selecting a *committee*, a small subset of the validator seats, to propose and confirm a transaction block. The selection process may be triggered periodically on a clock, after each update, or a combination. The randomness of the selection is cryptographically guaranteed, rather than relying on a trusted operator. The exact methods vary among protocols.

A common tool for committee selection is the *verifiable random function* (VRF) system. Every validator with some number of consensus seats assigned to a public key can evaluate a unique VRF on its own. The VRF output is a pseudorandom value, which indicates whether the public key has been selected to the committee. The probability the public key is selected is proportional to the fraction of seats it owns.

Nakamoto consensus (employed by Bitcoin) works very differently. In proof-of-work Nakamoto consensus the validators compete to solve puzzles uniquely tied to the block proposal and current state of the ledger. Validators

[7] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. *Consensus in the presence of partial synchrony.* J. ACM, 1988.

[8] Danny Dolev and H. Raymond Strong. *Authenticated algorithms for Byzantine agreement.* SIAMCOMP, 1983.

will generally accept the first block proposal they hear with a valid puzzle solution, as long as it builds on the longest chain of block proposals it has seen so far.

Proof-of-stake Nakamoto follows the same paradigm.[9] Validators are assumed to have weakly sychronized clocks and have a uniform method for generating timestamps. Similar to committee selection, a validator with a public key evaluates a unique VRF on the current timestamp, and if it obtains an output below a threshold determined by the number of seats it owns it will immediately broadcasts this along with a block proposal. Honest validators always extend the longest chain of consistent block update proposals they have seen so far. New proposals with either a very old timestamp, a timestamp in the future, or insufficiently spaced timestamps are rejected. Whether proof-of-work or proof-of-stake, this type of consensus fundamentally takes a longer time to converge on a common view but has other beneficial properties.

Corruption tolerance This is the maximum fraction of seats that may be controlled by corrupt validators before consistency or liveness would fail. (In stake or work permissioning models this is the fraction of stake or work controlled by an adversary). In an asynchronous network, corruption tolerance is always less than 1/3. In a strongly synchronous network, optimal corruption tolerance is possible, but requires a very slow protocol. There is generally a tradeoff between corruption tolerance and practicality. Also, any protocol that improves corruption tolerance above 1/3 in a synchronous network will inevitably worsen the corruption tolerance further below 1/3 in an asynchronous network.

Responsiveness This is how fast the consensus protocol confirms agreement on a ledger update. A protocol that can confirm a transaction block update at close to the actual speed of network communication has *instant finality*. There is a tradeoff between responsiveness and corruption tolerance. It is impossible to design a consensus protocol that guarantees instant finality if a 1/3 or greater fraction of the validator seats might be corrupt.[10] In a network where the maximum time delay Δ of messages is known, consensus protocols may tolerate up to 1/2 corruption and achieve a confirmation delay

[9] Rafael Pass and Elaine Shi. *The sleepy model of consensus.* Asiacrypt, 2017.

[10] R. Pass and E. Shi. *Hybrid consensus: Efficient consensus in the permissionless model.* DISC, 2017.

proportional to Δ.[11] *Optimistic responsivness* is a hybrid property between fast consensus with low corruption tolerance and slow consensus with high corruption tolerance, designed for a synchronous network.[12] A protocol that is optimistically responsive will enjoy instant finality under optimistic conditions (e.g. 3/4 of the validators are honest) and will enter a stage with slower confirmation times during worst-case conditions (e.g. almost 1/2 of the validators are corrupt).

Partition tolerance This is how the consensus protocol responds to network failures. The *synchrony model* defines the types of network failures considered, which may include nodes going offline, unpredictable communication delays, or complete partitions of the network. A network adversary who has infected the routers of honest nodes with malware might even reroute or entirely block messages based on their content. A list of synchrony models in order of increasing level of optimism (i.e. most realistic to least realistic assumptions) are as follows:

1. *Asynchronous.* Nodes can go offline arbitrarily or be partitioned permanently. Consensus protocols that satisfy consistency cannot guarantee liveness in an asynchronous network.

2. *Partial synchrony.*[13] Messages between honest nodes are eventually delivered, but there is no known upper bound on their delivery time. Consensus protocols that satisfy consistency cannot guarantee liveness in a partially synchronous network.

3. *Majority weak synchrony.*[14] At any given point in time a majority[15] of nodes are both honest and online (able to deliver messages to one another within a known maximum time δ). The network adversary may strategically select which nodes to disconnect at any given time, as long it does not exceed the majority of honest nodes. Disconnected nodes may not realize they are under attack and may still receive messages from the network adversary. In one variant the adversary may select

[11] S. Micali and V. Vaikuntanathan. *Optimal and player-replaceable consensus with an honest majority.* MIT CSAIL Technical Report, 2017; Rafael Pass, Lior Seeman, and Abhi Shelat. *Analysis of the blockchain protocol in asynchronous networks.* Eurocrypt, 2017.

[12] Rafael Pass and Elaine Shi. *Thunderella: Blockchains with optimistic instant confirmation.* Eurocrypt, 2018.

[13] Dwork et. al., *Consensus in the presence of partial synchrony.*

[14] Yue Guo, Rafael Pass, Elaine Shi. *Synchronous, with a Chance of Partition Tolerance.* Crypto, 2019.

[15] In a stake-based system this means a majority of the nodes weighted by stake.

the disconnected set based on the messages they are about to send. In a more optimistic variant, the adversary must choose the set of nodes whose network connections it controls before being able to inspect the messages they will send.

4. *Sleepy synchrony.*[16] Honest online nodes are able to communicate with each other within a known time delay, but some honest nodes may be *asleep*. A sleeping node is distinct from partitioned (disconnected) nodes in the weak synchrony model. A sleeping node is simply offline and will not accept messages even from the network adversary. For example, it is possible that the online validators participating in consensus only control 1% of the overall number of consensus seats in the system and 99% of the seats are in the hands of offline validators who may eventually decide to participate in consensus at at a later point in time. Any cosensus protocol that maintains liveness in the sleepy synchrony model is not even weakly partition tolerant, i.e. fails consistency even with majority weak synchrony.[17]

5. Δ-*synchrony.* There is a known maximum time delay Δ on messages sent between any two honest nodes.

6. *Strong synchrony.* Messages are delivered between any two nodes in the network almost immediately (i.e. at actual network speed, within seconds).

Adaptive security Bribery and targeted corruption is a concern in consensus protocols that select committees, despite the fact that each committee is selected randomly. It is much easier for the adversary to corrupt 1/3 of the randomly elected committee members at each time interval than to corrupt sufficiently many validators controlling 1/3 of the entire stake at one time. This is called an *adaptive adversary.*

To address adaptive corruption attacks, the protocol can be designed such that a newly elected committee member only broadcasts one message at the instant it was elected.[18] In a committee selection based on VRFs, each newly elected member discovers independently that it has been elected

[16]R. Pass and E. Shi, *The sleepy model of consensus.*

[17]Guo et. al.

[18]J. Chen and S. Micali. *Algorand: The efficient and democratic ledger.* Arxiv.org, 2016; Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, Nickolai Zeldovich. *Algorand: Scaling Byzantine Agreements for Cryptocurrencies.* SOSP, 2017.

to the committee and broadcasts this provable claim to the rest of the network. In the same message that reveals it was elected it also includes a vote. However, the adversary may still corrupt this member and cause it to send out other conflicting votes to disrupt the consensus protocol and potentially break consistency.

As a further mitigation, validators may generate fresh public keys for each committee selection process and immediately delete the private key used to sign a vote before broadcasting the vote. Validators generate and commit (i.e. post to the ledger) their public key for a given committee selection round ahead of time, before the VRF seed for that round becomes known. This separates the validator's temporary *signing key* for a given round from the validator's long-term *staking key*. This approach is still only adaptively secure if players honestly delete their keys, which is called adaptive security in the *erasures model*. Recently, a new stake-based consensus protocol called "consensus through herding" achieved adaptive security without assuming key erasures.[19] The protocol works very differently than other stake-weighted committee election protocols. It has a slow confirmation time and is only proven to have consistency and liveness in the Δ-synchronous model.

Accountability and elasticity ("slashing") What happens after corruption exceeds the tolerance threshold and the corrupt validators break consistency or liveness? Is the network doomed forever? One would hope the network could be more elastic, enabling honest nodes to reform a subnetwork that excludes the corrupt nodes. Although there is no way to prevent a consistency error once the corruption tolerance threshold is exceeded, once the honest nodes discover that there was an error can they hold accountable the corrupt validators that were responsible? Can they then eliminate the culpable validators from their view of the network?

Eliminating a culpable validator from a permissioned consensus protocol amounts to taking away all the consensus seats it controls. In a consensus protocol that uses stake-based permissioning, this is equivalent to erasing all the stake in a validator's accounts (called *slashing*). If the honest validators can agree that a validator violated the protocol then they can consistently slash the validator.

There are several challenges with this approach. First, validators may control consensus seats (i.e. stake) under many different identities, and only

[19]T.H. Hubert Chan, Rafael Pass, and Elaine Shi. *Consensus through herding*. Eurocrypt, 2019.

violate the protocol with one identity at a time. Even with a perfect slashing mechanism, it would take many rounds of repeated protocol violations before this validator's control is reduced and honest validators regain a supermajority of the seats. Second, it is circular to assume the honest validators may use the broken consensus protocol to agree on slashing the culpable validator. On the other hand, if there is irrefutable evidence that a validator is culpable then in theory slashing could be implemented without consensus. The honest validators would still need to agree that the evidence has propagated sufficiently before resuming the protocol. This either requires an agreement protocol or waiting a sufficiently long time Δ (in a Δ-synchronous network).

In some consensus protocols, such as Nakamoto consensus, validators could cause consistency errors without being caught and held culpable. However, in classical consensus protocols designed for 1/3 corruption in asynchronous networks every consistency error creates irrefutable evidence that a particular validator violated the protocol.[20] The Casper FFG protocol proposed by Ethereum Research uses a classical-style stake-weighted agreement protocol to finalize checkpoints in a ledger that is operated over an underlying Nakamoto-style consensus protocol.[21]

1.6 Alternative investments

The world of alternative investments is one of many areas of finance plagued by opaque money management and inefficient information tracking, including investment funds such as hedge funds and private equity funds, and peer-to-peer lending platforms.

[20]These protocols follow a two-phase voting paradigm in which validators first broadcast a *proposal vote* for proposed values (i.e. transaction blocks) and then broadcast a *confirmation vote* if and only if they observe a quorum of proposal votes (i.e. more than 2/3 proposal votes for the same value). Validators may change their proposal votes after a timeout period in which they never heard a quorum, but once a validator broadcasts a confirmation vote it may never broadcast a proposal vote for a conflicting value. The timeout period falls between *rounds* and votes are always submitted for a particular round. A consistency error will never occur unless some validator either sends conflicting proposal votes for the same round or sends any vote that conflicts with one of its own prior confirmation votes. Both of these protocol violations are detectable and the conflicting votes are irrefutable evidence.

[21]Validators must lock their stake in collateral accounts to participate in finalizing checkpoints. If a consistency error occurs, the protocol uses the resulting irrefutable evidence to slash the responsible validator. The protocol has never been implemented or deployed and is underspecified in terms of how it deals with the challenges discussed above.

An investment fund is operated by a fund manager (i.e. a general partner) who generally sets up a fund through established relationships with investors. The investors must trust the fund manager to manage the fund according to the rules and structure of the fund and faithfully employ the promised investment strategy. Otherwise, investors are exposed to issues like fraud, lack of transparency, and charter creep. Lending platforms match investors seeking high-interest loans with creditworthy borrowers. Similar to investment funds, the lending platform acts as an intermediary between investors and borrowers, and therefore must be trusted to vet borrowers and transfer capital honestly.

Even in highly regulated markets, information asymmetry abounds and investment funds and lending platforms remain highly opaque. In an increasingly complex financial system, scams and money laundering are rising concerns. This is particularly true in Asian markets, where a boom in online investment products over the last decade has been accompanied by a slew of high profile scams, such as the P2P lending platform Ezhubao or the Fanya Metal Exchange. Major fraud is not limited to Asia, and high-profile examples are plentiful. In the USA, Bernie Madoff defrauded over 4,500 investors of more than $60 billion, admitting to federal prosecutors in 2008 that the wealth management arm of his business was largely a ponzi scheme. As well, in June 2018, the Abraaj Group, a $14 billion Private Equity firm, filed for provisional liquidation due to broad-based compliance breakdowns.

Today, much of the investing process requires direct relationships and intermediaries, such as private banks or online platforms. These intermediaries play the role of matchmaker and aggregate investors to make raising money more efficient for fund managers. In return, these intermediaries typically get paid from both sides of the transaction.

A typical fund's back-office is complex and antiquated. Functions such as reporting, taxation, and compliance are largely paper based, requiring manual labor to check and ensure funds are operating within compliance and investors are receiving information in a timely manner.

Lastly, there exists no unified method or platform for distributing and exchanging fractional ownership in less liquid funds such as private investment funds. Any such system would require the capabilities we see in public markets including adequate transparency, cost efficiency, and tracking.

2 The Findora platform

Financial service applications on the Findora platform are built around instruments that live on digital distributed ledgers managed and secured by a global network. Applications will leverage tools for confidential financial transactions that balance privacy with proofs of regulatory compliance.

An example application supported by the Findora platform is a Smart Investment Fund (SIF), a fund that operates over smart contracts, introducing new levels of trust and transparency into investment funds while simultaneously respecting confidentiality. While a fund manager oversees the fund and dictates where money should flow, the tracking and recording of all assets is distributed across a network. The platform's privacy tools, which use special-purpose zero-knowledge proofs and multi-party computation, enable regulators and investors to ensure that the fund is compliant while preserving the confidentiality of fund participants.

2.1 Architecture

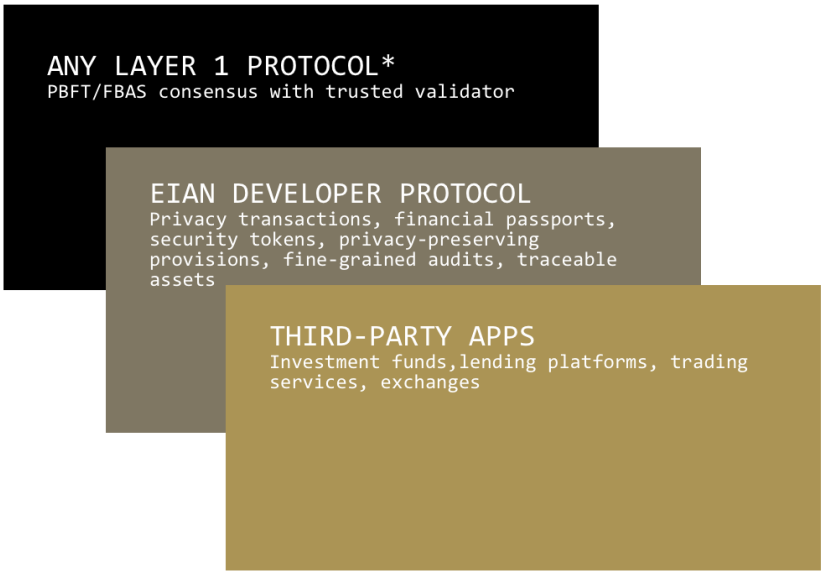


Figure 1: Findora platform layers.

The Findora platform is divided into three layers.

Financial service applications Financial service applications, called *Finapps*, sit at the topmost layer. Finapps can be freely developed and deployed by developers on the Findora network.

Findora developer tools The middle layer provides multi-asset issuance and transfer, financial passports (Findora's identity tool), audit and asset-tracking tools, and privacy-preserving compliance tools that use special purpose zero-knowledge proofs and multi-party computation.

Distributed ledger protocol The base layer is an underlying distributed ledger protocol that supports confidential payments, smart contracts, multi-signature account, and non-custodial exchange. The consensus and governance model for Findora utilizes Finsense consensus, while private and consortium side-ledgers may use their choice of consensus protocol.

2.2 Financial passports

Financial passports aggregate information about a user, starting with basic information about the user's accreditation and financial identity, to credit rating/scores, AML whitelisting, etc.

All the information in a user's passport is verified and signed by at least one authority. These signed statements are presented in the form of *cryptographic selective disclosure credentials*[22]. This means the information is authenticated in such a way that users can selectively reveal components of their identities without unnecessarily compromising the privacy of their entire personal financial profile. Furthermore, users can demonstrate complex statements about their authenticated profile (e.g. an income range, or threshold conjunction of several qualifications) without revealing any precise personal details at all.

Financial passports function much like the public key infrastructure of the Internet. There are a set of *root* authorities who are able to issue credentials to users and also endorse other sub-authorities with the privilege to issue credentials. Users can then store and manage their own credentials just as they would manage their cryptographic keys, and they may also elect to share or store their credentials with a third party custodian.

[22]J. Camenisch and A. Lysyanskaya. *Signature schemes and anonymous credentials from bilinear maps.* Crypto, 2004; Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn and George Danezis. *Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers.* NDSS 2019.

Ideally, the public keys of the root authorities would be managed on a common ledger of information and recognized by everyone participating in the network. In reality, there may be several subnetworks that each recognize their own custom list of root credential authorities. For example, one circuit of banks may recognize only the US SEC as a root authority whereas another circuit of banks recognizes only the China SRC as a root authority. This does not outright prevent asset-flow between the two subnetworks, but introduces friction as the users of each sub-network could be concerned about running afoul of their local security exchange laws. This friction can be resolved through intermediaries or cross endorsements by the respective root authorities.

2.2.1 Selectively disclosable identity forms

A selectively disclosable identity form contains the following items:

- `credPubKey`: public key for which the user knows a private key
- `attributeList`: list of self-claimed attributes provided by the user
- `attributeBitVector`: bit vector indicating attributes the identity provider approves
- `providerSignature`: - signature from the identity provider on the other form items

The identity form may also contain signatures from multiple identity providers, in which case the form content is structured as follows:

```
body {
   credPubKey
   attributeList
}

signatures {
   List <providerPubKey , attributeBitVector ,
   providerSignature >
}
```

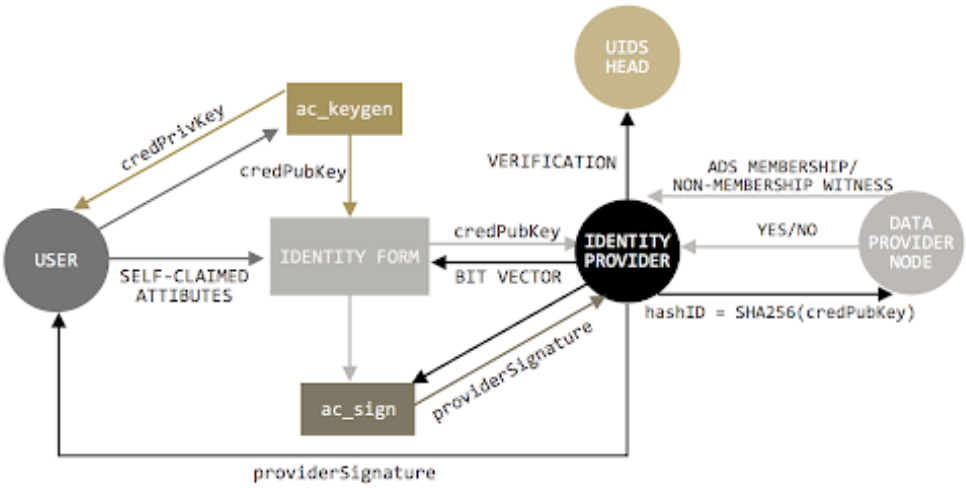
The identity form is never publicly revealed as this would compromise anonymity. A special selective reveal protocol is described below that uses zero-knowledge proofs. The "public key" and signature on the identity form are not simply any standard public key and digital signatures. They are crafted in a special way using pairing-based techniques for cryptographic anonymous credentials.

This is to enable efficient zero-knowledge proofs in the selective reveal and verification protocol.

The public key is also randomizable, which means that there is a function to generate a fresh public key (unlinkable to the old public key) but with the same secret key. The public key acts as a cryptographic commitment to the secret key and the randomized public key is a fresh commitment to the same secret key.

The key functions used for selectively disclosable identity forms are `ac_keygen`, `ac_sign`, `ac_reveal`, `ac_verify`, and `ac_randomize_pubkey`. The implementation details are described under **Implementation and Optimizations** below.

2.2.2 Obtaining a new identity form



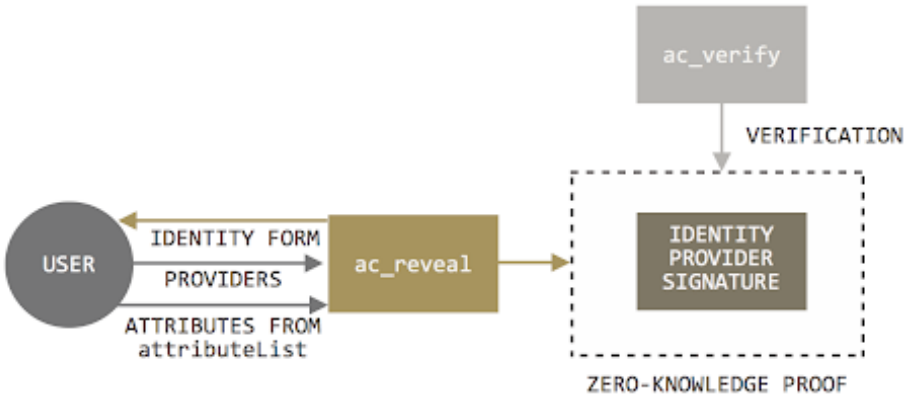
1. The user generates a new anonymous credential public key `credPubKey` using the `ac_keygen` function. This also generates a `credPrivKey` that is stored in the user's wallet.

2. The user fills out an identity form that includes `credPubKey` and a list of self-claimed attributes. The attributes include information about the user's identity and/or credentials. The user sends this to an identity provider.

3. The identity provider appends a bit vector to the identity form that is the same length as the `attributeList`. The identity provider checks each attribute and sets the corresponding bit to 1 if it approves the attribute or 0 if it rejects the attribute. The identity provider then creates a signature `providerSignature` using the function `ac_sign` with the identity form as input. The identity provider sends this back to the user.

4. The user stores the completed and signed identity form. The user may repeat this process with other identity providers.

2.2.3 Selective reveal and verification



1. The user selects which attributes from `attributeList` to reveal. It passes these as inputs to `ac_reveal` function along with the rest of the completed and signed identity form. If the identity form contains signatures from multiple providers then the user selects the providers that have approved the attributes being revealed.

2. The `ac_reveal` function generates a zero-knowledge proof that the identity form contains a signature from the specified identity provider on the given attributes. The proof demonstrates that the identity provider has signed a form including a bit vector indicating approval of the given attributes and also the user's `credPubKey`. The proof does not reveal `credPubKey`, but demonstrates the user's knowledge of the

corresponding secret key (i.e. it is infeasible to generate this proof without the secret key for `credPubKey`).

3. The `ac_verify` function verifies the proof generated by `ac_reveal`.

2.2.4 Linking identity to accounts and registry

Users may link an identity credential to a ledger address. The Findora ledger contains an address identity registry that users may update via ledger transactions. The user takes a provider signatures `sig` from the identity form, computes a randomization of the signature `rsig = ac_randomize(sig)`, and adds the entry (*addr, rsig*) to the identity registry.

The credential is randomized before it is added to the identity registry so that the link is not publicly revealed. The randomized signature still binds the ledger address to a unique identity form, similar to a cryptographic commitment. When the owner of address is required to reveal selective identity information (e.g. as part of a transaction) linked to the address, the user generates the selective reveal proof for the identity form with the provider signature `sig`. The owner additionally provides a zero-knowledge proof that the `rsig` in the ledger registry is a randomization of `sig`.

2.2.5 Implementation and optimizations

The implementation of the functions `ac_keygen`, `ac_sign`, `ac_reveal`, `ac_verify`, and `ac_randomize` are described here. These implementations use the Pointcheval-Sanders signature and anonymous credential protocol.

Public parameters The public parameters specify two prime numbers p and q, two cyclic elliptic curve groups G_1,G_2 of prime order p, an order p subgroup G_T of the extension field $GF(q^{12})$, and a bilinear pairing function $e : G_1 \times G_2 \rightarrow G_T$. This is based on the BLS12-381 construction of pairing-friendly elliptic curves. The prime q is about 384 bits and the prime p is about 256 bits. A hash function Hash is a collision-resistant hash function like SHA256 that has the properties of a random oracle.

Issuer public key setup The function `ac_keygen` has two modes, one for an Issuer and one for a User. The Issuer is the party that signs credential forms with `ac_sign`. The first output of `ac_keygen`("Issuer") is a secret key vector of n 256-bit integers:

$$SK_1 = (x, z, y_1, ..., y_n)$$

The integer n is the maximum number of distinct attributes the issuer can sign on an identity form. The second output is a public key constructed as follows:

$$g_1 \leftarrow G_1, g_2 \leftarrow G_2, X_1 = g_1^x, X_2 = g_2^x, Z_1 = g_1^z, Z_2 = g_2^z, Y_i = g_1^{y_i}, Y_{n+i} = g_2^{y_i}$$

$$PK_1 = (g_1, g_2, X_2, Z_1, Z_2, Y_{n+1}, ..., Y_{2n})$$

Initiating a credential request In the user mode, `ac_keygen`("User") outputs a randomly generated 256-bit secret key sk and credential public key `credPubKey` = Z_1^{sk}. A user requesting an attribute vector $\vec{\alpha} = (\alpha_1, ..., \alpha_m)$ from the Issuer sends `credPubKey` and $\vec{\alpha}$ to the Issuer.

Issuer completes a credential request The Issuer runs the `ac_sign` function on a `credPubKey` and approved list of credential attributes $\alpha_1, ..., \alpha_m$. The steps of `ac_sign` are as follows:

1. Compute $C :=$ `credPubKey` $\cdot \prod_{i=1}^{m} Y_i^{\alpha_i}$
2. Sample a random 256-bit integer u
3. Output $\sigma = (\sigma_1, \sigma_2)$ where $\sigma_1 = g^u$ and $\sigma_2 = (X_1 C)^u$

The output is the signature sent back to the User. Importantly, the user does not know the secret value u, which binds the signatures on all the individual attributes and on `credPubKey` together.

The computational cost of `ac_sign` is a multi-exponentiation of size m and two additional exponentiations in G_1, or approximately $2 + \frac{m}{logm}$ exponentiations.

Credential reveal The User chooses a subset $S \subseteq \{1, ..., m\}$ of the attributes to reveal. The User then executes `ac_reveal` on `credPubKey`, sk, and the signature σ. This function uses a collision-resistant hash function H such as SHA-256. The steps of `ac_reveal` are as follows:

1. Sample random 256-bit integers r and t
2. Compute $\sigma' = (\sigma'_1, \sigma'_2) = (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$
3. Compute a sigma proof of knowledge of private inputs t, sk, and public inputs $\{\alpha_j\}_{j \notin S}$ such that the following verification equation holds:

$$e(\sigma'_1, X_2) \cdot e(\sigma'_1, g_2^t) \cdot e(\sigma'_1, Z_2^{sk}) \cdot e(\sigma', \prod_{i=1}^{m} Y_{n+i}^{\alpha_i}) = e(\sigma'_2, g_2)$$

The computation of this sigma proof consists of the following steps:

1. Choose random 256-bit integers $\beta_1, \beta_2, \{\gamma_j\}$ for $j \notin S$
2. $\pi_1 \leftarrow g_2^{\beta_1} \cdot Z_2^{\beta_2} \cdot \prod_{j \notin S} Y_{n+j}^{\gamma_j}$
3. $c \leftarrow H(\pi_1)$ and $\pi_2 \leftarrow (ct + \beta_1, c \cdot sk + \beta_2, \{c\alpha_j + \gamma_j\}_{j \notin S})$

The total computational cost to `ac_reveal` is approximately 3 exponentiations in G_1 and a multi-exponentiation in G_2 of size $k = m - |S| + 2$, which requires approximately $k/logk$ total exponentiations in G_2.

Credential verify The function `ac_verify` verifies the proofs in the output `ac_reveal`. The steps of the function on the input (σ', π_1, π_2) are:

1. Parse π_2 as (ρ_1, ρ_2, ξ_j) and derive c= Hash(π_1)
2. Verify that:

$$e(\sigma'_1, \pi_1^-1 \cdot X_2^c \cdot g_2^{\rho_1} \cdot Z_2^{\rho_2} \cdot \prod_{j \notin S} Y_{n+j}^{\xi_j} \cdot \prod_{j \in S} Y_{n+j}^{c \cdot \alpha_j}) = e(\sigma'_2, g_2^c)$$

The total computational cost to `ac_verify` is approximately 2 executions of the pairing function, an exponentiation in G_2, a multi-exponentiation of size $m + 3$ in G_2, approximately the cost of $\frac{m+3}{log(m+3)}$ exponentiations in G_2, and 1 multiplication in G_T.

Batch verification of credentials The verifier receives many randomized signatures (P_1^i, P_2^i) and proof tuples(π_1^i, π_2^i) and $c_i \leftarrow Hash(\pi_1^i)$ for i = 1 to K distinct outputs of `ac_reveal` on possibly distinct `credPubKey` values belonging to different users.

Referring to the values(ρ_1, ρ_2, ξ_j) from the proof π_2^i, let $J_i i$ denote the large multi-exponentiation:

$$J_i := (\pi_1^i)^{-1} \cdot X_2^{c_i} \cdot g_2^{\rho_1} \cdot Z_2^{\rho_2} \cdot \prod_{j \notin S} Y_{n+j}^{\xi_j} \cdot \prod_{j \in S} Y_{n+j}^{c_i \cdot \alpha_j}$$

An individual verification of each credential would require computing J_i and all the individual pairings $e(P_2^i, g_2^{c_i})$ and $e(P_1^i, J_i)$.

Using the technique of random linear combinations to batch verify K credential signatures, `ac_batch_verify` chooses random coefficients $r_1, ..., r_K$ and verifies instead the single equation:

$$e(\prod_{i=1}^{k}(P_2^i)^{c_i r_i}, g_2) = \prod_{i=1}^{k} e(P_1^i, J_i)$$

This involves computing $K + 1$ pairings, K multiplications in G_T , K multi-exponentiations in G_2 of size m to compute each J_i, and a multi-exponentiation of size K in G_1. The amortized cost for large K is just 1 pairing, $\frac{m}{logm}$ exponentiations in G_2, $\frac{1}{logK}$ exponentiations in G_1, and 1 multiplication in G_T per credential. For large K this is dominated by the cost of 1 pairing and $\frac{m}{logm}$ exponentiations in G_2.

2.3 Digital asset tokens

Findora supports the creation and distribution of secondary digital asset tokens representing economic ownership as well as the myriad of complex relationships between investors, fund managers, and companies. For example, an asset token may represent fractional ownership of an investment fund and the liabilities of the fund participants.

A *digital asset token* is an immutable and transferable representation of value, analogous to a physical token such as a ticket or dollar bill that can be passed from hand to hand. Tokens may be generated, destroyed, or transferred. However, digital tokens are an abstraction, and there are different ways in which they can be represented and transferred digitally. A *digital security token* is a special kind of digital token that represents a liability and carries meaning under some real world jurisdiction. For example, a token could represent the ability of one counterparty (token holder) to sue the other (token issuer). This is distinct from a digital asset like Bitcoin, which is akin to commodity money (like gold) and has intrinsic value ascribed by the network of its holders.

Digital tokens that represent legal tender, percent ownership of a finite resource, or transferable liabilities such as capital commitment must prevent double spending. If Alice transfers a digital dollar token to Bob, then she cannot transfer the same digital dollar token to Charlie. The double spending problem can be solved using a common ledger of information that tracks creation, ownership, transfer, and destruction of all digital tokens.

In other words, there is a data structure for maintaining a shared database among all participants and a consensus mechanism for reaching agreement on the current state of the database. This database may be hosted by a

single provider (centralized database), a fixed set of providers (consortium database), or in the case of Findora, hosted in a decentralized form (public blockchain). Each solution has its own benefits and drawbacks. For this reason, the Findora developer stack is being designed with modularity in mind, such that it can be configured both for Findora, a public network, as well as on top of a variety of alternative ledger instantiations.

2.3.1 Anchors

The issuer of a digital asset token is called the anchor. For instance, a bank might anchor a token representing reserve-backed assets. A fund manager might anchor a token representing fractionalized ownership in the fund's holdings. Bitcoin and Ethereum are examples of tokens used as digital currencies that have a decentralized issuance and do not have a specific anchor. A token standard form is used by an anchor to define a token class, specifying the token function. Findora provides a new Smart Asset Framework (SAF), a programmable standard interface for financial assets. The SAF encompasses diverse financial assets, from class A shares to real estate investments.

Anchored asset tokens may be transferred peer-to-peer and digitally tracked, but they are only valuable to the token holder if the token holder has a legal (or otherwise trust-based) relationship with the token anchor. For example, a bank-anchored fiat token (i.e. CBDC, central bank digital currency) represents a bank's fiduciary promise to honor its tokens one-to-one with a unit of fiat money, offering the token holder the ability to exchange tokens for fiat money at any point in time at a fixed exchange rate. Similarly, a Bitcoin holder must rely on the Bitcoin network to maintain the integrity and value of Bitcoins.

Ideally, an asset token holder should be able to transfer its token to another party who does not share a legal/trust relationship with the same anchor. In this case, a security token seller cannot simply transfer ownership of the token to a buyer as the token does not legally represent any value for the buyer in its current form.

This is solved by identifying liability paths, similar to the way paths are identified in credit networks for interbank transfers. Suppose Alice has a security token from Anchor X and she is matched in an order book with Bob who does not have a relationship with Anchor X, but has a relationship with a Anchor Y. Furthermore, both X and Y have relationships with Anchor Z. In this case, the following transfers and issuances would be executed atomically:

1. Alice transfers to Anchor Z her security token from Anchor X
2. Anchor Z issues its own security token to Anchor Y
3. Anchor Y issues its own security token to Bob

Intermediaries can be incentivized to facilitate trades via transaction fees or percentages of returns. Exchange services at Findora's application layer can provide order books that efficiently determine the optimal path between any two counterparties engaging in a trade (e.g. using metrics such as least cost), and execute trades along that path instantly and atomically.

2.3.2 Asset token data model

A basic record of asset token ownership in Findora is a triple of three values: ($address$, $amount$, $asset\_type$)

The $address$ is a public key, and the owner of the asset must know the corresponding private key. The $asset\_type$ uniquely references the asset definition, i.e. what this asset token represents on the Findora ledger. When an anchor issues a new asset token on Findora, it must first define the asset token by creating an *asset class*. The asset class is a data form containing all the information about the asset's legal meaning, from its broad classification (e.g. promissory note, bond, stock, debt, real estate) to its finer contractual details. This can include policies restricting its ownership and transfer. The asset class includes the public key of the anchor and is digitally signed by the anchor. Every asset class defined on the Findora ledger has a unique *asset code*, which by default is a 32-byte content-addressable hash of the data form contents. This asset code serves as the unique $asset\_type$ reference in a record of ownership.

Fungible assets may have many indistinguishable units of the same $asset\_type$ (e.g. units of stock, bank issued money, etc). In this case, the $amount$ field in a record of ownership is used to indicate that an $address$ owns multiple units of the same $asset\_type$. For a non-fungible asset (e.g. real estate) there is only one unit in existence of a given asset class, hence $amount = 1$.

Asset token creation Before an asset token can be issued to any ledger public key address, its asset class must be defined on Findora's ledger. The high level structure of an asset token class is as follows:

- Code: Unique asset code representing this asset on the ledger (16 bytes)

- Digest: Content addressable hash of the static token data
- Issuer: Includes at a minimum the public key of the anchor, plus optionally a description of the issuer, such as a linked verified identity
- Memo: Description of the asset's legal meaning and classification, e.g. bank note, debt, stock, promissory note, property.
- Confidential memo: A private memo, used for confidential assets
- Updatable: A flag indicating whether the asset issuer can update the memo fields. All updates are append-only operations and memo history is tracked.
- Units: The total number of (public) units in existence, not applicable to all asset types. This field is not part of the static data and is updated every time the asset issuer mints new units in a transaction.
- Confidential units: Similar to units, but keeps track of the total number of confidential units in existence, hidden inside a cryptographic commitment. This is helpful for confidential asset issuances.
- Policies: Rules that restrict and regulate transactions involving this asset token. Assets with policies are called smart assets.

Asset token issuance and transfer Asset issuance is a transaction, digitally signed by an anchor, that creates a new asset record of ownership. It assigns units of a given asset type to a ledger address. The digital signature must validate against the issuer public key that is specified in the corresponding asset class definition. Newly created asset records are called *valid*, and they can be invalidated by future transactions. The ledger keeps track of all valid/invalid records.

Asset transfers are transactions that transfer ownership of existing asset units between two ledger addresses. The simplest asset transfer consumes (i.e. invalidates) an existing asset record and create a new record that changes only the *address*, but keeps the same *amount* of units and *assettype*. More generally, an asset transfer may consume multiple existing valid asset records, and creates new asset records that redistribute the units among new addresses. As a basic rule, an asset transfer transaction must consume as many units of a given asset type as it creates. However, asset transfers must also respect the custom policies specified in the asset class.

3 Confidential asset transfers

A confidential asset transfer is a transaction that transfers ownership of an asset from one address to another, but hides the details of the asset that is transferred. In the case of basic asset transfers, this includes the *amount* and *asset_type* fields in the input and output asset records consumed and created during the transaction. More advanced confidential transactions in Findora also hide the identities of addresses in transactions. In particular, addresses can be fully anonymized each time a party transacts so that a user's transactions are unlinkable.

In order to explain how the most basic confidential transfers work in Findora, let us take a closer look at the anatomy of a Findora asset transfer transaction.[23] An asset transfer is executed simply by posting a *transfer note* to the Findora ledger, denoted `XfrNote` for short.

```
pub struct XfrNote{
  pub(crate) body: XfrBody,
  pub(crate) multisign: XfrMultiSig,
}

pub struct XfrBody{
  pub(crate) inputs: Vec<BlindAssetRecord>,
  pub(crate) outputs: Vec<BlindAssetRecord>,
  pub(crate) proofs: XfrProofs,
}
```

The `XfrBody` contains a list of input asset records and output asset records. For confidentiality these asset records are *blinded*, using cryptographic *commitments*. These are implemented using Pedersen commitments over an elliptic curve group called "Ristretto". We call the blinded record data struct a `BlindAssetRecord` to distinguish it from a plain `AssetRecord`.

```
pub struct AssetRecord{
  pub(crate) amount: 64,
  pub(crate) asset_type: Option<[u8;16]>,
  pub(crate) public_key: XfrPublicKey, // ownership address
}

pub struct BlindAssetRecord{
  pub(crate) asset_type: Option<[u8;16]>,
  pub(crate) amount_commitment: CompressedRistretto,
  pub(crate) asset_type_commitment: CompressedRistretto,
  pub(crate) blind_share:  CompressedEdwardsY,
```

[23] Technically, Findora transactions bundle operations and an asset transfer in Findora is a single operation.

```
  pub(crate) lock_amount: ZeiCipher,
  pub(crate) lock_type: ZeiCipher,
  pub(crate) public_key: XfrPublicKey,
}
```

The `lock_amount` and `lock_type` are encryptions of the asset record's *amount* and *type* fields respectively. They are encrypted under `public_key`, the public key of the asset record owner (i.e. recipient address).

The cryptographic commitments are perfectly hiding, which makes them distinct from encryptions. They do not contain any information that can be decrypted by someone with a key. Rather, they serve as a hidden fingerprint for the committed information, similar to how a server can send a hash of a file before sharing the file. The hash is a unique fingerprint that can be measured from the file, but the file cannot be obtained from the hash. Cryptographic commitments can only be "opened" or "unblinded" given the unique information that was committed and a secret value called the *blinding factor*. If C is a commitment to a message m using blinding factor r, then C can be uniquely computed from m and r, and r is proof that C was a commitment to m. In Findora's blinded asset records, the blinding factors are shared with the new asset owner (i.e. transfer recipient) using a method similar to a Diffie-Hellman key exchange. This user derives the blinding factors from `blind_share` and its private key corresponding to `public_key`. The user needs these blinding factors in order to check that the decrypted contents of the record are correct (i.e. as approved by the validators) and will also need to use them to transfer ownership of the asset in a future transaction.

```
pub struct OpenAssetRecord{
  pub(crate) asset_record: BlindAssetRecord,
  pub(crate) amount: u64,
  pub(crate) amount_blind: Scalar,
  pub(crate) asset_type: AssetType, //type AssetType = [u8
    ;16]
  pub(crate) type_blind: Scalar,
}
```

The `XfrProofs` contain a zero-knowledge proof that the blinded output records are valid with respect to the blinded input records. Specifically, it proves that the sum of output amounts for each asset type in the output records equals the sum of input amounts for the same asset type in the input records. To be more precise, if there are n inputs records and m output records with the following variables defined:

- α_i is the amount in the ith input record

- β_j is the amount in the jth output record
- $\mathsf{In}[t]$ is the set of input indices with asset type matching t
- $\mathsf{Out}[t]$ is the set of output indices with asset type matching t
- $\mathcal{T}$ is the complete set of types among the output records

then `XfrProofs` prove that:

$$\text{For all } t \in \mathcal{T} \sum_{i \in \mathsf{In}[t]} \alpha_i = \sum_{j \in \mathsf{Out}[t]} \beta_j$$

Finally, an `XfrNote` is only valid if each input `BlindAssetRecord` correctly references an existing valid record on the ledger created by a previous transaction. Thus, the transaction that encapsulates an `XfrNote` must also include references to transfer notes in previous transactions. These references are called transaction output sequence ids (TxoSIDs).

```
pub struct AssetTransferBody {
  pub inputs: Vec<TxoSID>,
  pub transfer: Box<XfrNote>,
}
```

Full-node validators that have access to the entire ledger use each ith TxoSID to look up a `BlindAssetRecord` in a previous `XfrNote` output and check that it matches the ith `BlindAssetRecord` in the current transaction's `XfrNote`. Validators also check that TxoSID is still valid. Once a TxoSID is used in a transaction it becomes invalid (i.e. the record is "consumed").

3.1 Transaction commitments and proofs

3.1.1 Math background

Finite groups The cryptographic protocols used for confidential transfers in Findora require as a tool a finite group of prime order where certain computational problems are hard. A finite group $\mathbb{G}$ is a finite set with a defined group operation on elements in the set. We use the "+" symbol to denote operations between a pair of group elements. Operating on any two elements in the set gives another element in the set. There is a unique "0" element such that $0 + g = g$ for any $g \in \mathbb{G}$. Every element g has an inverse element denoted $(-g)$ such that $g + (-g) = 0$. The order of the group is the number of elements in the set. The integers under addition modulo n are a simple example of a group of order n. This groups is denoted $\mathbb{Z}_n$ and contains all the integers less than n. The integers coprime with n are a group

under integer multiplication, denoted $\mathbb{Z}_n^*$. The set of integers $\{0, ..., p-1\}$ for prime p are the group $\mathbb{Z}_p$ under addition and also the group $\mathbb{Z}_p^*$ under multiplication if we exclude 0. Groups with this property are called finite fields, and this field is denoted $\mathbb{F}_p$.

Elliptic curve groups More advanced groups can be constructed by looking at *points* on curves defined over a finite field. An elliptic curve E over $\mathbb{F}_p$ is defined by an equation of the form $y^2 = x^3 + ax + b \bmod p$, where $a, b \in \mathbb{F}_p$. The elliptic curve group $\mathbb{G} = \mathrm{E}(\mathbb{F}_p)$ consists of all points $(x, y) \in \mathbb{F}_p$ that satisfy this equation, and there is a group operation that interpolates any two points to find a third point on the curve. Findora uses the curve called Curve25519, which uses the prime number $p = 2^{255} - 19$ and the curve equation $y^2 = x^3 + 486662x^2 + x$.

Quotient groups Yet another type of group, called a *quotient* group, can be built on top of an existing group $\mathbb{G}$ that has a special kind of subgroup $N \subset G$ called a *normal* subgroup. A subgroup N is a subset of $\mathbb{G}$ that is also a group under the same operations, and N is normal if for all $g \in \mathbb{G}$ and $h \in N$ the element $g + h + (-g)$ is contained in N. In a commutative group where the order of operations doesn't matter every subgroup is normal. Given the normal subgroup N and $\mathbb{G}$, the quotient group $\mathbb{G}/N$ is built by forming a partition of $\mathbb{G}$ into subsets called *equivalent classes*. Here two elements $a, b \in \mathbb{G}$ are placed in the same equivalence class if and only if $a - b \in N$. These equivalence classes are the new elements on the group $\mathbb{G}/N$ and are represented by picking an element from each equivalence class to "represent" the class. If $\bar{a}$ and $\bar{b}$ are two representative elements, the group operation finds the representative element $\bar{c}$ for $c \in \mathbb{G}$ where $c == \bar{a} + \bar{b}$. If $\mathbb{G}$ has order m and N has order n, then $\mathbb{G}/N$ has order m/n, which is always an integer.

Ristretto group Findora uses the Ristretto group, which is a quotient group built from the elliptic curve group on Curve25519. The elliptic curve group on Curve25519 has order $8p$ for the prime:

$$p = 2^{252} + 27742317777372353535851937790883648493$$

The Ristretto quotient group is built from a normal subgroup of order 8 therefore has prime order p.[24]

[24]For more information on the Ristretto group we refer the reader to **https://ristretto.group/**

Abstract notation For the purpose of describing cryptographic protocols, we will use the following notations for operations in the Ristretto group. We use $\mathbb{G}^p$ to denote the Ristretto group. Elements in $\mathbb{G}^p$ (represented by points on Curve2559) are denoted with capital letters, e.g. $A, B \in \mathbb{G}^p$. Lower case letters are used to denote elements in $\mathbb{F}^p$, also called "scalars".

1. Group addition: $C \leftarrow A + B$ is the group operation in $\mathbb{G}_p$, taking two representative curve points A, B and returning a third representative curve point C.

2. Scalar multiplication: aC denotes the element of $\mathbb{G}_p$ obtained by adding together a copies of C using the group addition operation, where $a \in \mathbb{F}_p$ is interpreted as a positive integer less than p.

Discrete logarithm problem (DLP) An algorithm that solves DLP in the group $\mathbb{G}_p$ is given a non-zero element $\mathrm{G} \in \mathbb{G}_p$ and a random element $H \leftarrow_R \mathbb{G}_p$ and succeeds to output $a \in \mathbb{F}_p$ such that $aG = H$ with non-negligible probability. DLP is widely believed to be computationally hard in the Ristretto group, i.e. with current computing cababilities there is no efficient algorithm for DLP.[25]

Decisional Diffie Hellman (DDH) A prime order group has the DDH security property if it is computationally difficult to distinguish the tuple (aC, bC, abC) from the tuple (aC, bC, rC) for randomly chosen $a, b, r \in \mathbb{F}_p$ and any element $C \neq 0$. The Ristretto group is widely believed to have the DDH property with current computing capabilities. The DDH property is a stronger security assumption than DLP because solving DLP in $\mathbb{G}_p$ breaks the DDH property.

3.1.2 Pedersen commitments

A Pedersen commitment is a group element $C \in \mathbb{G}_p$ that is cryptographically binding to a scalar $m \in \mathbb{F}_p$, but completely hides m. The scalar m can encode an arbitary message by using any suitable cryptographic collision resistant hash function Hash: $\{0,1\}^* \rightarrow \mathbb{F}_p$ that maps data strings onto the field $\mathbb{F}_p$.

The element C is generated in a unique way using m and an additional random scalar r, called the blinding factor. Therefore, given m and r, it is easy to verify that the Pedersen commitment C is the correctly generated

[25]However, the DLP will be efficiently solvable in any group (including Ristretto) if quantum computing ever becomes practical.

output. The Pedersen commitment generated from m and r is computationally binding to the message m as long as DLP is hard in $\mathbb{G}_p$. The ability to find alternative inputs m' and r' for which the Pedersen commitment generates the same point C leads to an efficient solution for DLP in $\mathbb{G}_p$.

Pedersen commitments have the following components:

1. PedersenSetup: G, H are randomly generated "base points" in the group $\mathbb{G}_p$.
2. PedersenCommit$(m) \rightarrow (C,\ r)$: The input is $m \in \mathbb{F}_p$ and the output is $mG + rH$ for a random $r \leftarrow_R \mathbb{F}_p$.
3. PedersenOpen $(C, r, m) \rightarrow (r, m)$: The opening of a commitment C are a pair of values m, r for which it can be verified that $mG + rH = C$.

Homomorphic addition Two Pedersen commitments $C_1 = m_1G + r_1H$ and $C_2 = m_2G + r_2H$ can be added using the group operation in $\mathbb{G}_p$ to form the new commitment $C_3 = C_1 + C_2$. The element C_3 is a commitment to $m_1 + m_2 \bmod p$ with the blinding factor $r_1 + r_2 \bmod p$.

Proving commitment equality To prove that two Pedersen commitments $C1$ and $C2$ commit to the same scalar value $m \in \mathbb{F}_p$ we can use what is known as a Schnorr proof. The Schnorr proof is a zero-knowledge proof of knowledge of a discrete logarithm. Given $C = aG$, a Scnorr proof demonstrates that the prover (i.e. the algorithm generating the proof) must have an a such that $aG = C$, and yet the proof does not reveal any additional information about C.

Note that if $C_1 = mG+rH$ and $C_2 = mG+r'H$ then $C_1-C_2 = (r-r')H$. The zero-knowledge proof that $C1$ and $C2$ commit to the same value is simply a Schnorr proof of the discrete logarithm of $C_1 - C_2$ base H, i.e. it proves knowledge of the value r^* such that $C_1 - C_2 = r^*H$. If the prover knows the values (m, r, r') to open both $C1$ and $C2$ to m then it knows $r^* = r - r'$ and is able to succeed in creating this proof. On the other hand if the prover were able to open C_2 to (m', r') such that $m' \neq m$ and still succeed in this zero-knowledge proof protocol, then it must know the value $z = (m - m')(r^* - r + r')^{-1}$ such that $zG = H$. In other words, this prover would efficiently solve the DLP in $\mathbb{G}_p$. Therefore, as long as DLP is computationally hard in $\mathbb{G}_p$ this cannot happen. The Schnorr proof for public inputs $A = aH$ and H with private input $a \in \mathbb{F}_p$ contains two scalars (z, c) formed as follows:

1. Randomly sample $r \leftarrow_R \mathbb{F}_p$
2. $c \leftarrow \mathsf{Hash}(A, H, rH)$
3. $z \leftarrow ac + r \bmod p$

The proof $\pi = (z, c)$ is verified by checking that $c = \mathsf{Hash}(A, H, zH - cA)$.

Batch equality proofs Given commitments $C_1, ..., C_n$, proving that each C_i commits to the same scalar value m is equivalent to proving that each C_i commits to the same value as C_1. This reduces to proving knowledge of the discrete log of each $D_i = C_1 - C_i$ base H. There is an optimization to avoid producing n separate Schnorr proofs for each D_i. Assume $D_i = a_i H$. Given scalars $\beta_1, ..., \beta_n$ the prover knows the discrete log of $D = \sum_{i=1}^{n} \beta_i D_i$ base H, which is the value $a = \sum_{i=1}^{n} \beta_i a_i$. Moreover, if these scalars are chosen randomly after the commitments $C_1, ..., C_n$ are declared then a prover that does not know the discrete log of at least one D_i base H will fail to solve the discrete log of D base H with all but negligible probability. It is important that the β_i are independently random and cannot be arbitrary. For instance if $D_0 = m_0 G + r_0 H$ and $D_1 = m_1 G + r_1 H$ such that $\beta_0 m_0 + \beta_1 m_1 = 0$ then the prover knows that $\beta_0 D_0 + \beta_1 D_1 = (\beta_0 r_0 + \beta_1 r_1) H$.[26]

The batch equality proof uses the Fiat-Shamir heuristic to derive the coefficients βi using a hash function. Assuming the prover knows $C_i = m_i G + r_i H$, the steps for producing the batch equality proof are:

1. Set $\beta_i = Hash(C_1, ..., C_n, i)$.
2. Compute $D = \sum_{i=1}^{n} \beta_i (C_i - C_1)$
3. Provide a Schnorr proof for D base H, i.e. $D = aH$, using knowledge of $a = \sum_i \beta_i (r_i - r_1) \bmod p$.

The size of the proof is a single Schnorr proof, i.e. two scalars. The verifier of this proof also performs the first two steps above to derive D from the inputs using the defined hash function and then verifies the Schnorr proof.

Range proof (Bulletproof) A range proof is a zero-knowledge proof that a Pedersen commitment C commits to a scalar m that as an integer is bounded in a range $[L, R]$, i.e. $L \leq m \leq R$. Findora's confidential transfers require a range proof that $m \in [0, 2^{64}]$. Our range proofs are implemented

[26] This informal description only provides the intuition for why the protocol is secure. There is a formal proof of security for this protocol.

using Bulletproofs[27], a specialized zero-knowledge proof system. Unlike zk-SNARKs, Bulletproofs do not rely on a trusted and complex setup. Bulletproofs are particularly suited for range proofs on small ranges: the proof for a 64-bit range is less than $1KB$ and takes only milliseconds to both create and verify. Bulletproofs have a batching mode where a range proof from m points is only 64 $\log(m)$ bytes larger than a range proof for a single point (e.g. a batch proof for 100 points is less than 500 bytes larger). Bulletproofs also have a batch verification mode where the amortized time to verify many range proofs is approximately 0.34 ms per proof.

3.2 `BlindAssetRecord` and `XfrProofs`

Now that we have introduced the building blocks, we will describe in more detail the commitments in `BlindAssetRecord` and the zero-knowledge proofs in `XfrProofs`.

The `amount_commitment` and `asset_type_commitment` are Pedersen commitments over the Ristretto group $\mathbb{G}_p$. The recipient's public key is also an element $P \in \mathbb{G}_p$ with a corresponding private key scalar s such that $P = sG$. The element G is the same public base that is specified in the Pedersen commitment setup parameters.

When a user creates the blind asset record in a transaction the user samples a random scalar $k \in \mathbb{F}_p$ called the blinding key and derives the blinding factors $r_0 \leftarrow Hash(kP, 0)$ and $r_1 \leftarrow Hash(kP, 1)$. The `blind_share` is the element kG. The recipient (i.e. asset record owner) can derive $kP = skG$ using the secret key s and the `blind_share`, and therefore may recover the blinding factors.

The structure of `XfrProofs` is:

```
pub struct XfrProofs {
  pub(crate) range_proofs: Option<BatchRangeProof>,
  pub(crate) equality_proof: EqualityProof,
  pub(crate) asset_proof: Option<BatchEqualityProof>
}
```

Range proof on outputs The `range_proofs` are a batched Bulletproof range proof that all the amount commitments in the output blind asset records of the transfer transaction are Pedersen commitments to 64-bit integers. With m outputs this proof has a size of approximately $700 + 64 \log(m)$ bytes.

[27]Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. *Bulletproofs: Short proofs for confidential transactions and more.* In Bulletproofs: Short Proofs for Confidential Transactions and More, p. 0. IEEE, 2018.

Equality proofs Given the asset amount commitments $C1, ..., Cn$ on all the input blind asset records and the asset amount commitments $C_1', ..., C_n{}'$ on all the output blind asset records, define $C = \sum_{i=1}^{n} C_i$ and $C' = \sum_{i=1}^{m} C_i'$. Let $T_1, ..., T_n$ denote the asset type commitments of the inputs and $T_1{}', ..., T_m{}'$ prime denote the asset type commitments of the outputs. The `equality_proof` contains a Pedersen equality proof that C and C' commit to the same scalar value. The `asset_proof` contains a batch Pedersen equality proof that all T_i and T_i' commit to the same value.

3.3 Smart contracts

Smart contracts are dynamic programs driven by transactions on a distributed ledger. The smart contract state is also stored on the ledger. In particular, smart contracts can be used to encode complex financial relationships between counterparties with a lifetime spanning more than a single one-time transaction. In essence, smart contracts define rules that determine how the contract's state is affected by transactions on the ledger and how this state may in turn trigger transactions. For programming smart contracts, a basic opcode instruction set solves a software consensus problem: contracts written in arbitrary code can lead to ambiguity, and different validators may claim different interpretations. A network that is synchronized on a set of basic opcodes serves as a "virtual machine" for programming more complex contracts. This is particularly important in Ethereum where anyone in the network may be a validator. Ethereum boasts a "Turing complete" opcode set, meaning that any arbitrary program can be built out of the opcode set.

However, financial contracts are in general specialized programs that do not come close to requiring a Turing complete language to encode. Furthermore, when a contract is enforced by only a relatively small subset of the network, then custom software consensus is much more feasible. Findora takes the view that custom opcodes for a particular contract need only be distributed to a custom set of validators of that contract. Participants in a contract will be able to determine for themselves whether or not they trust a threshold majority of the contract validators. The custom validators for this contract issue multi-signature transactions based on the result of their independent validation protocol. This enables "Turing complete" contracts to operate over an underlying distributed ledger that only supports basic multi-signature contracts.[28]

[28] A similar idea is employed in Hyperledger and Corda.

Smart assets and smart accounts *Smart assets* are special cases of a smart contract that attach terms to specific asset tokens, and regulate the transfer of these assets with policies. These are discussed more in the next section on security token terms. *Smart accounts* are stateful accounts, i.e. hold balances of different assets and other dynamic data, but additionally have policies that restrict how transactions update the state of the account. Account policies are Boolean functions that take as input the current state of the account and a new transaction. Validators check that the account policy returns true on any transactions that updates the state of a given account and reject the transaction when the policy returns false.

Since transactions may bundle together several operations (e.g. affecting multiple accounts at once), account policies can be used to "trigger" other transactions in response to an account update by only approving transactions that bundle together the appropriate operations (e.g. requiring updates to the account balance to be bundled with an automatic fee to an operator). *Smart contracts* generalize smart accounts even further as they are dynamic stateful programs that are stored on the ledger and run arbitrary code to determine state transitions. As explained above, smart contracts may designate a targeted set of custodians who approve state changes to the contract in the form of a threshold multi-signature. These multi-signature contracts can run arbitrary code as the main network validators only need to verify the signature. Finally, Findora features *native smart contracts* that do not designate a custodian and are processed by the entire network of validators.

3.3.1 Native smart contracts

An important distinction in Findora is between *stateful* and *stateless* smart contracts. Stateful contracts are more expressive than stateless contracts, but run the risk of placing a high computational and storage burden on the network validators. A stateless contract is a special kind of smart contract that does not have any dynamic state. Instead, the smart contract program defines a Boolean function that is run on a bundle of transaction inputs. If the program returns true then the bundle of transaction inputs are executed atomically and appended to the ledger's transaction log. Policies on asset tokens and smart assets are examples of stateless smart contracts. The final transaction bundle also references the smart contract program, which is a content addressable static data object on the Findora ledger. Verifying the transaction bundle requires running the contract program on the bundle and checking the result of the contract program.

Stateless smart contract functionality can also be achieved using *variables*

and *preconditions* on normal transaction operations. For many use cases, placing variables and preconditions on transactions is sufficient to simulate the desired smart contract behavior entirely through native transaction operations. Stateless contracts are much better for scalability and are encouraged when stateful contracts can be avoided.

Variable addresses and preconditions Variables and preconditions can be defined in transactions. An operation can specify a variable destination address $[x]$, for example "transfer 100 USD from address A to $[x]$", and then a list of preconditions that the variable $[x]$ must satisfy for the operation to become valid. The preconditions may include:

- Ops: Other operations involving the same address [x]. These operations must be included in the same transaction's list of operations.
- Records: Asset records that are included as inputs to the transaction. The precondition may require a specific asset record and specify completely the contents of this record, or more generally leave some of the asset record fields as variables and specify a policy that the input asset record must satisfy.
- Policy: A policy applies to the variables referenced in the ops or input records. Supported native policies include a credential requirement on address variables in the ops or records, and arithmetic expressions on transfer amount variables. Evaluating the policy may require auxiliary inputs, such as a credential proof or other privacy-preserving compliance proof.

In pseudocode, an example operation with a variable address $[x]$, variable amount $[y]$, and preconditions is:

```
Operation A {
Transfer from addrA to [x] 300 USD tokens
preconditioned on
  Op: Transfer from [x] to addrA [y] SPCX tokens
  Policy: A credential proof that [x] is accredited
  Policy: y >= 1
}
```

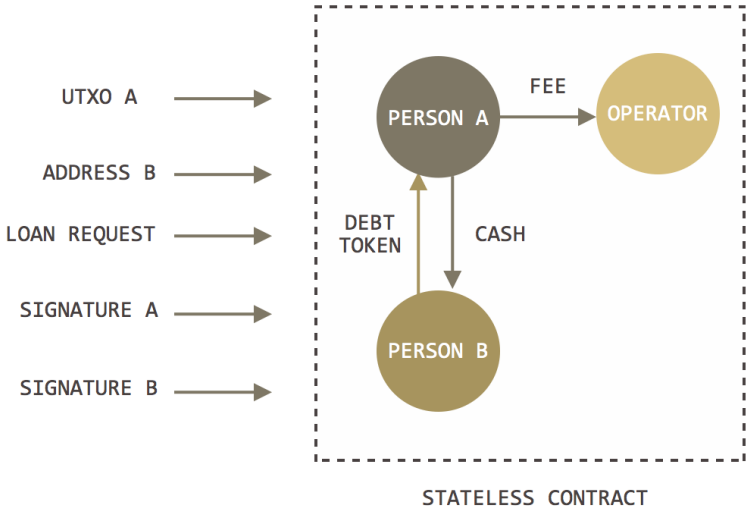
The validator processes transactions with variables and preconditions by scanning the list of operations and substituting variables until all the preconditions on each transaction are satisfied. For the above example, suppose that a new Operation B was added to the transaction operation list:

```
Operation B  {
Transfer from addrB to [x] 1 SPCX tokens
preconditioned on
  Op: Transfer from [x] to addrB [y] USD tokens
  Policy: y > 275
<proof of addrB accreditation>
}
```

When a validator processes Operation A and Operation B together, substituting for the variables, this resolves to:

```
Transfer from addrA to addrB 300 USD
Transfer from addrB to addrA 1 SPCX
```

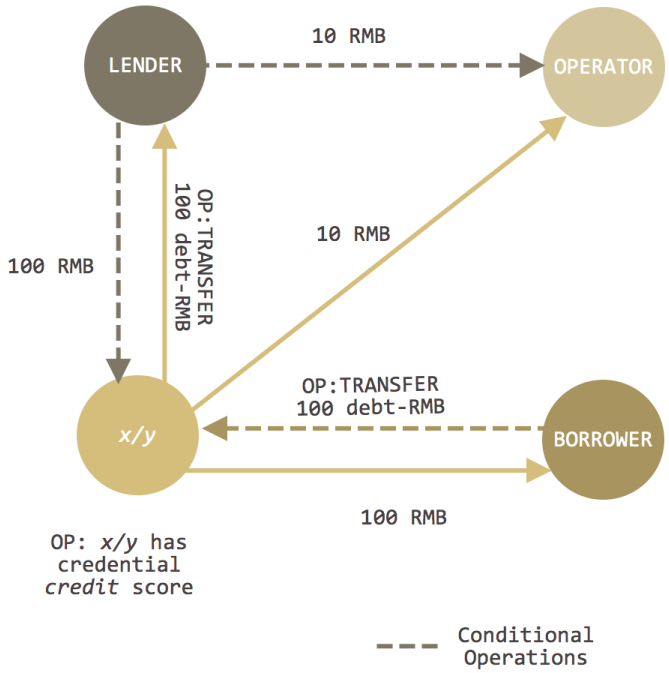
Example 1: Stateless contract for loan matching A stateless contract can be used to implement basic lending platform functionality.



The contract first checks if the loan request matches the loan pledge, the signatures are valid, and the input record (i.e. UTXO A) has sufficient funds to cover the loan. If all conditions are met, then it (1) pays from address A to address B, (2) transfers a 1% fee to the operator, and (3) issues the debt token from B to A. The transactions that are issued are distinct from normal transactions, as the final transaction is "issued" by the smart contract itself. Verifying one of these transactions requires checking the result of the contract program. The final transaction includes the contract inputs, the contract output, and references the smart contract program, which is a content addressable data object that can be looked up on the ledger. Since

there is no intermediate state, only the final contract-issued transactions affect permanent ledger storage. The verification just checks the inputs, runs the program, and checks the validity of the outputs.

Example 2: Loan matching with variables and preconditions Transactions with variables and preconditions can be used to implement a simple loan matching functionality for a lending platform. These avoid the need to define any smart contract program.



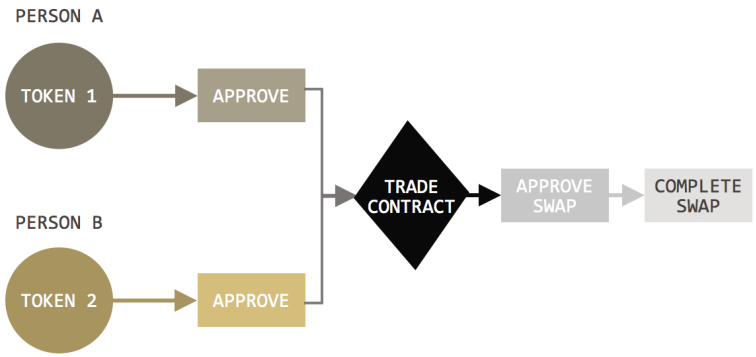
The loan offer is a signed transfer operation that contains a transfer from address A to variable address [x] of 100 USD and a transfer of 10 USD as a fee to the operator address C, both preconditioned on a second operation that would transfer 100 debt-USD tokens from [x] to address A, and a credential proof that [x] has a sufficient credit score. The debt-USD token contains terms regarding a loan repayment schedule with interest.

A loan request is a signed transfer operation submitted to the network that transfers from address B to a variable address [y] 100 debt-USD, preconditioned on a second operation that would transfer 100 USD to address B. The transfer operation contains in its memo field a credential proof that address B belongs to a borrower with a sufficient credit score.

The lending platform operator collects both the loan request and the loan offer signed operations and packages them together into a valid transaction.

When packaged together, the preconditions of both operations are satisfied. Once validated by the network, this transaction will send 100 USD from address A to address B, send 10 USD to the operator, and send 100 debt-USD tokens from address B to address A.

Example 3: Atomic swaps An atomic swap is a ledger transaction that allows two parties to trade two different tokens such that there is no risk of one party defaulting.



In this transaction, the tokens for both person A and person B are approved and used as inputs into a trade contract that transfers from A to B a token of type 1 (that A is guaranteed to have) and from B to A a token of type 2 (that B is guaranteed to have). A and B use signatures as proof that they own these tokens. When the contract is executed, an atomic swap takes place as both transactions in the the contract are executed (A to B of token type 1 and B to A of token type 2), allowing both parties to exchange two different tokens at minimal risk of default.

3.4 Security token terms

Terms that restrict the transfer of security tokens can either be encoded universally in the definition of a token or enforced on a custom basis through smart contracts. Terms of the first kind are processed by the validators of the base layer ledger protocol. For example, a security token issuer could place a credential requirement in the definition of the token stipulating that all token holders must have accreditation approval from a specified set of KYC authorities. Any transaction transferring this token from one party to another would require a special signature from the recipient demonstrating this credential.

Custom-enforced terms would be more appropriate for assets in an investment fund, which are governed according to the investment terms. These terms obviously vary depending on the type of fund, and investors in the same fund may also have different terms. For instance, in private investment funds, terms can regulate and restrict transfers, govern lock ups, or whitelist authorized investors. In mutual funds, terms can restrict investment to specific asset classes, limit risk, or prescribe cash balances.

3.5 Regulators

In financial markets, participants are held accountable by regulators who both set and enforce rules across the system. The role of market regulator demands a delicate balance between the privacy of participants and regulators' ability to monitor and enforce existing regulations. As a result of this tension, regulators bring enforcement actions against fund managers on an inconsistent basis and outright fraud is often not caught until it reaches a noticeable scale. The Findora platform frees regulators from this balancing

act. By utilizing zero knowledge proofs, selective disclosure identity, and privacy preserving computation, it is able to deliver on both privacy and transparency simultaneously. These cryptographic techniques unlock new possibilities for more effective regulation that were previously not possible, without compromising user privacy. Functionally, this means that funds can cryptographically prove to regulators that they are operating compliantly (e.g. by using proofs-of-solvency), without having to disclose details of their actions or investments.

3.6 Privacy-preserving compliance tools

One of the cornerstone values in the design of Findora is allowing for various levels of transparency while preserving confidentiality. For example, this is what enables Smart Investment Funds (Section 6) to offer regulators visibility into general fund information (assets, holdings, investor credentials) while simultaneously keeping confidential other selected fund-related data (investing activity, investors, terms etc.). Advanced cryptography excels precisely at this task of simultaneously achieving transparency and confidentiality.

The two most relevant cryptographic techniques that we use are *zero-knowledge proofs* and *multi-party computation*. A zero-knowledge (ZK) proof is a technique for showing that a statement is true without revealing any further information other than the statement's validity. ZK proofs can also

be used to prove *knowledge* of a secret, such as the password to unlock an account, without revealing the secret itself. Similar to ZK proofs, secure multi-party computation (MPC) enables a set of parties who each hold a private input to a computation to jointly learn the output of the computation on the inputs without revealing to one another any additional information about the private inputs. For example, MPC could be used to run a secret-bid auction without relying on a trusted party to collect the bids.

In general, MPC either requires many rounds of interaction or expensive computations such as fully-homomorphic encryption, and is thus impractical for smart contracts to implement. However, SIFs will use specialized efficient MPC protocols that are sufficient for Findora's use cases (for example, keeping the balance sheet of a fund confidential).

Confidential payments A basic confidential payment is a transaction denominated in some token unit (e.g. a currency like Bitcoin) that transfers a hidden amount from one address/account to another in such a way that the system as a whole can still publicly verify the validity of the transaction (e.g. that the total number of token units in existence were preserved in the process).

Confidential asset transfer Confidential asset transfers use cryptographic commitments to hide the details of the assets held in the sending and receiving account (e.g. type and balance) and uses zero-knowledge proofs to demonstrate that these commitments were updated correctly in accordance with the rules of the asset transfer, e.g. that the sum of the new balances equals the sum of the old balances and neither balance has gone negative. When the asset type is kept confidential the proof must also show that balances were updated in the two accounts under the same asset type identifier without revealing this identifier. Our implementation uses a combination of techniques including Pedersen commitments, ElGamal encryption, Bulletproofs[29], and Σ-Bullets[30]. These proofs take only milliseconds to generate and verify.

[29]B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille,. and G. Maxwell.*Bulletproofs: Short Proofs for Confidential Transactions and More.* https://eprint.iacr.org/2017/1066.pdf.

[30]B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. *Zether: Towards Privacy In a Smart Contract World.* 2018.

Proofs of solvency A proof of solvency[31] demonstrates that the value of asset-backed tokens owned by an entity such as a fund or exchange exceeds its liabilities (e.g. to investors). This tool is particularly relevant when the assets held by the entity producing the proof are confidential, i.e. hidden with cryptographic commitments. The generic technique behind proofs of solvency takes a set of accounts or transactions labeled liabilities and a set of accounts labeled assets, and produces a zero-knowledge proof of knowledge of the secret keys controlling the asset accounts and that the sum total of these asset balances (weighted by type) exceeds the sum total of liability balances.

Proof of whitelisted assets This is a proof that the identifier of a confidential asset involved in a transaction is contained in a whitelist set that does not reveal the identifier itself. For example, the whitelist could be maintained in a Merkle tree and the proof is a zero-knowledge inclusion proof of this identifier in the tree.

Balance range proofs These use Bulletproofs to prove a range on the balances contained within accounts or transactions, such as a minimum balance on an investment account or an upper bound on the amount transferred in a transaction.

Capability-specific viewing keys Regulators can be given keys to decrypt a user's account or transaction content that does not enable them to issue transactions on the user's behalf (i.e. a read-only key and not a signing key that enables writes). Viewing keys can also be attached to proofs of compliance, such as proofs of solvency or whitelisted assets. These keys can reveal more granular information to authorized regulators than the result of the proof, but still without revealing all the details of individual accounts. If trusted hardware execution environments are available then these can also be used to implement highly fine-grained capability-specific viewing keys.[32]

Confidential multi-source payment While a confidential transfer hides (from the public) the amount transferred in a transaction, this amount is always revealed to the recipient. Consider a multi-party payment from multiple independent sources to a recipient. Hiding from the recipient the amounts

[31] Dagher, Bünz, Bonneau, Clark, and Boneh. *Provisions: Privacy-Preserving Proofs of Solvency for Bitcoin Exchanges*, 2015. http://www.jbonneau.com/doc/DBBCB15-CCS-provisions.pdf.

[32] Ben A Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. *Iron: Functional encryption using Intel SGX.* CCS 2017.

transferred from each source and revealing only the aggregate amount can be accomplished with a simple MPC from *linear secret sharing*[33]. This is a two round protocol. In the first round each source splits its input into linear secret shares, one for each other source. In the second round, the sources compute an aggregate sum over the shares they have received from other sources and publicly post the result of their local computation. The final aggregate sum can be derived from these inputs. To keep this final aggregate private from the public, the recipient can participate in the secret sharing scheme as well so that only the recipient can unmask the final sum. Moreover, if the underlying confidential payments use a *homomorphic commitment scheme* (such as Pedersen commitments) then the MPC can be efficiently modified so that the recipient is guaranteed to know whether or not the final sum it learns is correct.

Confidential asset tracer This tool enables an asset issuer to track its assets and see the details of all transactions involving these assets even if they are confidential to the public. This may be important, for instance, to a company issuing equity that wants to be able to see all the shareholders in the company at any given point in time. Token transfer terms combined with zero-knowledge proofs enable the creation of assets that retain this tracer without requiring interaction with the issuer. The issuer may come online at any point in time and decrypt the details of all transfers that contain this tracer.

Privacy-preserving computation Other examples of privacy-preserving computations that can be implemented with MPC include sealed-bid auctions and order-book matching, where the values of bids and asks are hidden until matched in order to prevent front running. These computations will require more than the extremely lightweight two-round secret sharing techniques described above.

State of the art MPC protocols for general functions take several approaches to reducing the computational costs and rounds of interaction that constrain practical deployment. One method is to push most of the computation and interaction into a "pre-processing phase" that distributes setup information between the involved parties in preparation for an "online phase" where the parties repeatedly execute an efficient privacy-preserving compu-

[33]A k-of-n linear secret sharing of a value x splits it into n parts [x]1,...,[x]n such that any k - 1 shares reveal no information at all, yet any k shares are sufficient to reconstruct the value x.

tation on private inputs. For instance, the SPDZ-BMR protocol[34] has an "online phase" consisting of only three rounds of interactions and minimal computation. This optimization is appropriate for a scenario in which a fixed set of participants will repeatedly engage in a privacy preserving auction or other fund-related computation over a period of time.

Another approach is to shift the responsibility of maintaining privacy onto a distributed set of "third party" servers such that privacy is kept as long as there is at least one "honest" non-colluding server. More efficient MPC protocols have been developed[35] in this model, known as the *server-aided model*. A natural candidate for these servers would be the distributed set of validators of a custom smart contract. However, it is important to note the difference between trusting that the validators will not secretly collude to expose private information versus visibly violating the integrity of a contract.

4 Findora base layer

The base layer refers to the core functional layer of Findora that supports Findora's public financial ledger. The major operating agents in Findora's base layer are validators, gateways, archivers, and data providers. Gateways provide an entry point for users to send transactions. (Transactions are any requested modifications to the Findora ledger, issued using Findora's transaction API). Validators gather and verify transactions broadcasted from gateways, and agree on the confirmed transactions via the underlying consensus protocol. Data providers ensure that public users have authenticated query access to the ledger. Archivers ensure that historical ledger data is never forgotten.

The validators of Findora's main ledger run the Finsense consensus algorithm to propose and confirm transactions to the ledger. The ledger itself is a public database of financial records which is accessible by all nodes in the network. Records may include both encrypted and unencrypted content, and different parties (such as regulators or consumers) may have asymmetrical information about the ledger content, such as keys to decrypt certain content on the ledger. All transactions updating these records are immutably

[34]Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. *Efficient constant round multi-party computation combining BMR and SPDZ.* In Crypto 2015, pages 319–338.

[35]Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. *Secure multiparty computation goes live.* In Financial Cryptography, 2009.

recorded. The Findora platform also provides the capability for off-ledger verifiable storage of account information. The ledger data storage is fault-tolerant and permanent.

4.1 Authenticated distributed ledger

At the core of the base layer is a shared *authenticated data structure (ADS)*[36] that stores a set of accounts, a current state for each account, and the history of updates to the ledger. Each account references a list of type/balance pairs where 'type' is an asset description and 'balance' describes the asset balance in the account. Every account on this ledger is associated with a public key. The ADS has a tag representing its current state and a history of tags for every update. There is a procedure for generating a verifiable proof, called a *membership witness*, that any arbitrary portion of the ledger's history is consistent with this tag. An ADS has several efficiency advantages for the distributed setting:

1. Verifying that all nodes in the network agree on the state/history of the ledger only requires comparing short data tags.
2. Users (who do not store the whole ledger) can retrieve a portion of the ledger from a single node in the network along with an authentication proof that it is correct and consistent with the same ledger replicated on every other node in the network.
3. Users can submit updates to the ledger (i.e. a transaction) and obtain a short certificate from any node in the network that the updates were incorporated into the new state of the ledger.

There are many ways to build an ADS, each with their own nuances and advantages. Findora's base layer utilizes a new ADS based on cryptographic accumulators[37] that eliminates storage and memory bottlenecks and improves throughput by completely decoupling consensus and storage.

4.2 Confidential transactions

A transaction submitted to the ledger protocol batches one or more account operations to be executed atomically. The signature required for each operation in the transaction must also sign a digest of the entire transaction.

[36]Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi. *Authenticated Data Structures, Generically.* POPL, 2014. https://www.cs.umd.edu/ mwh/papers/gpads.pdf.

[37]Dan Boneh, Benedikt Bünz, and Ben Fisch. *Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains.* Crypto, 2018.

This guarantees that individual operations within the transaction cannot be individually replayed (non-atomically).

Transactions may also include *preconditions*, a logical expression whose input depends on the state of the ledger. For example, a precondition could evaluate whether a certain amount of time has elapsed between transactions. The precondition must resolve to true for the transaction to be valid. The same transaction can be broadcasted several times until the preconditions are met. This is what enables mutually distrusting users to atomically link their transactions.

A basic *confidential transaction* hides the value exchanged in a transaction. Confidentiality is a distinct privacy goal from anonymity, which also hides the identities of the accounts involved in the transaction. Confidential payments were introduced for the Bitcoin ledger model using a combination of Pedersen commitments and zero-knowledge proofs.[38] Findora implements a scalable solution for both privacy goals (confidentiality and anonymity). Importantly, Findora integrates identity with confidential transactions so that even anonymous addresses are uniquely bound to an identity/credential, which can be selectively revealed. This way viewers (e.g. regulator vs public users) of the ledger may have different levels of visibility into the identities of transacting parties depending on the access keys they have been given.

4.3 Multi-signature accounts

A multi-signature account is an account that is controlled by a distributed set of owners and every update to the account, whether a deposit or withdrawal of funds, requires a threshold multi-signature (TMS) from the owners. A TMS is a technique commonly used by blockchain smart-contracts. If there are n agents managing an account, then each agent holds a secret key that it uses to contribute to the TMS without revealing the secret. A basic k-of-n TMS is valid if and only if at least k out of the n agents contribute to the signature. More complex TMS validation logic is also possible. A weighted TMS assigns a weight to a signature from each agent and requires the weighted sum of signatures to exceed a threshold. More generally, any logical predicate could be used to define signature validity. While the size of most TMS signatures scales with the size of the set of agents, certain TMS

[38] G.Maxwell. Confidential transactions. https://people.xiph.org/ greg/ confidential$_v alues.txt$, 2016.

signatures (e.g. based on BLS signatures) can be made compact even when the set of n agents is large.[39]

5 The Findora network

Findora is a global public financial infrastructure. Maintained by an open and public set of validators, Findora is secured by Finsense, a novel consensus mechanism that combines both institutional trust and economic stake to guarantee honest behavior. Validators are anticipated to come from a diverse set of geographic regions, financial institutions, and industries. Participants and users of Findora will host, manage, and use various sets of applications that can seamlessly interact with each other.

Findora is also the central network which offers interoperability between private and consortia ledgers, in the same way that the Internet connects organizations' intranets. Findora validators which manage asset and data transfers between Findora and such side-ledgers are called *side-ledger interfaces*. Side-ledger interfaces explicitly define restrictions on the types of assets, transactions, identity providers, etc. that are allowed to cross into their side-ledgers.

Findora is a powerful system that will enable:

- Efficient cross border payments
- Open banking
- Global access to any capital market
- Transparent financial services

5.1 Financial Infrastructure Network Units

The native digital cryptographically-secured utility token of the Findora platform (FRA) is a transferable representation of attributed functions specified in the protocol/code of the Findora platform, which is designed to play a major role in the functioning of the ecosystem on the Findora platform and intended to be used solely as the primary utility token on the platform.

[39]Dan Boneh, Manu Drijvers, and Gregory Neven. *Compact Multi-signatures for Smaller Blockchains.* https://eprint.iacr.org/2018/483.pdf

FRA is a non-refundable functional utility token which will be used as the economic incentives which will be consumed to encourage users to contribute and maintain the ecosystem on the Findora platform, thereby creating a win-win system where every participant is fairly compensated for its efforts. FIN is an integral and indispensable part of the Findora platform, because without FRA, there would be no incentive for users to expend resources to participate in activities or provide services for the benefit of the entire ecosystem on the Findora platform. Users of the Findora platform and/or holders of FRA which did not actively participate will not receive any FRA incentives.

In Findora, network stakeholders hold and utilize Financial Infrastructure Network Units (FRA). Validators who hold FRA in eligible escrow accounts are proportionally selected in the FIN-weighted leader selection mode of Finsense consensus. Validators are thus able to collect transaction fees proportional to the number of FRA tokens they have staked. Finally, FRA must also be staked by side-ledger interface nodes wishing to move assets between the FRA and private/consortium side-ledgers.

The Distributor which issues and sells FRA shall be an affiliate of the Foundation. In particular, it is highlighted that FRA: (a) does not have any tangible or physical manifestation, and does not have any intrinsic value (nor does any person make any representation or give any commitment as to its value); (b) is non-refundable and cannot be exchanged for cash (or its equivalent value in any other virtual currency) or any payment obligation by the Foundation, the Distributor or any of their respective affiliates; (c) does not represent or confer on the token holder any right of any form with respect to the Foundation, the Distributor (or any of their respective affiliates), or its revenues or assets, including without limitation any right to receive future dividends, revenue, shares, ownership right or stake, share or security, any voting, distribution, redemption, liquidation, proprietary (including all forms of intellectual property or licence rights), right to receive accounts, financial statements or other financial data, the right to requisition or participate in shareholder meetings, the right to nominate a director, or other financial or legal rights or equivalent rights, or intellectual property rights or any other form of participation in or relating to the Findora platform, the Foundation, the Distributor and/or their service providers; (d) is not intended to represent any rights under a contract for differences or under any other contract the purpose or pretended purpose of which is to secure a profit or avoid a loss; (e) is not intended to be a representation of money (including electronic money), security, commodity, bond, debt instrument, unit in a collective investment

scheme or any other kind of financial instrument or investment; (f) is not a loan to the Foundation, the Distributor or any of their respective affiliates, is not intended to represent a debt owed by the Foundation, the Distributor or any of their respective affiliates, and there is no expectation of profit; and (g) does not provide the token holder with any ownership or other interest in the Foundation, the Distributor or any of their respective affiliates.

The contributions in the token sale will be held by the Distributor (or their respective affiliate) after the token sale, and contributors will have no economic or legal right over or beneficial interest in these contributions or the assets of that entity after the token sale. To the extent a secondary market or exchange for trading FRA does develop, it would be run and operated wholly independently of the Foundation, the Distributor, the sale of FRA and the Findora platform. Neither the Foundation nor the Distributor will create such secondary markets nor will either entity act as an exchange for FRA.

5.2 Finsense

Finsense is Findora's consensus algorithm, which enables a stable, high-throughput, and public network secured by both real-world trust anchors and stake. Finsense uses FRA tokens to represent consensus seats. The total supply of FRA tokens is large enough so that in theory anyone in the world could participate in consensus, however, individual validators may own more than one seat. A validator's influence in the protocol is proportional to the number of seats it holds.

Given a fixed distribution of FRA tokens, the protocol operates much like other proof-of-stake (PoS) protocols. Periodically, a small committee of seats are randomly selected to propose and confirm a new transaction block. The randomness of the selection is cryptographically guaranteed, rather than relying on a trusted operator. The technical method is based on selecting random committees at fixed intervals using a cryptographic VRF, as in other[40] stake-weighted consensus protocols. Each committee then runs a fast BFT protocol designed to work in a weakly synchronous network.

Validators who hold seats on a committee earn rewards by collecting transaction fees. A validator is selected to participate in a committee with frequency proportional to the number of FRA tokens may be exchanged among validators; however, there will be a lockup period after a validator

[40] Algorand,PiLi

participates on a committee before it may transfer the FRA tokens that contributed to its selection.

So long as the FRA-based consensus protocol appears to be operational and secure, honest[41] validators leverage the protocol itself to process all FRA transfers. However, if honest validators detect a security break or the FRA-based consensus gets stuck then these validators fall back to an FBA system to help resolve the error. In this sense the consensus protocol consists of two "channels": a main PoS channel and a secondary FBA channel that is activated only when the PoS channel fails.

5.2.1 Consistency, liveness, and finality

Finsense favors consistency over liveness. Consistency ultimately impacts the security of assets in the network more than liveness. Finsense does not assume the network will be Δ-synchronous at all times. This is an unrealistic assumption for any practical value of Δ. Therefore, the protocol will favor maximum corruption tolerance in an asynchronous network.

If Finsense were designed to tolerate greater than 1/3 corruption in weakly synchronous conditions this would reduce the corruption tolerance in asynchronous conditions. Therefore, Finsense just has a single fast consensus path that tolerates up to 1/3 corruption, even under asynchronous conditions. In addition to maintaining consistency in asynchronous conditions while corruption does not exceed 1/3 of the stake, the protocol will also involve a recovery process in the event that the threshold is exceeded. The recovery process cannot prevent the inevitable consistency errors that may occur when corruption exceeds the threshold, but will work to eliminate corrupt validators and enable the honest validators to regain more than 2/3 stake control.

Finsense maintains liveness under *2/3-weak-synchrony* network conditions. This is a network in which at least 2/3 of the validators (weighted by stake) are online, honest, and able to communicate with each other within a small network time delay. The protocol will be perfectly responsive when live. In other words, it confirms transactions with instant finality unless the network conditions are too unsafe.

[41] Honest validators are those who follow the prescribed protocol correctly.

5.2.2 Adaptive security, accountability, and recovery

Adaptive security Finsense discourages adaptive attacks in a different way than previous protocols. In our protocol, we use a cryptographic mechanism to ensure that a validator who signs two conflicting messages in a voting round with the same elected public staking-key leaks the corresponding private staking key in the process. This allows anyone in the network to submit a transaction stealing all FRA tokens that the validator's staking key controls. A corrupt validator might attempt to immediately transfer the FIN tokens to a new public key, but still risks losing the tokens due to race[42] conditions on the transfer. This gives validators a strong incentive against accepting bribes from an adaptive adversary to submit a conflicting vote. The value of the adversary's bribe to corrupt a staking key must at least exceed the value of all FRA tokens the staking key currently controls.

The cryptographic mechanism we use is called a *one-time signature*. Suppose a validator has a staking key mpk with corresponding private key msk, which control FRA tokens on the ledger. In a given voting round i, the validator uses its staking private key msk and a special derivation process to create a one-time signature key opk_i, along with a proof that opk_i was derived correctly. The proof can be verified against mpk. For each mpk, there is only one unique opk_i that can be derived in round i. Signing a message with opk_i requires the private key msk. Two opk_i signatures on distinct messages leaks msk.

Accountability and recovery There are only two kinds of consensus disruptions: consistency errors that cause honest validators to disagree and liveness errors that stall the system's progress. Honest validators eventually detect either type of disruption and activate the FBA channel to resolve it. In particular, validators use the FBA channel to agree on manual changes to FRA balances in an attempt to weed out corruption. Validators who commit *detectable violations* of the consensus protocol are held accountable. A detectable violation occurs when a corrupt validator approves invalid transactions or signs conflicting consensus votes. Only detectable violations can cause consistency errors. Honest validators will erase the FRA balances of validators who commit a detectable violation. In order to ensure that detectable violations are costly, the protocol imposes a minimum staking balance on each validator public key account. This prevents validators from

[42]Furthermore, there is a significant delay before other honest validators will accept the transfer from a staking key that participated in a round of voting, i.e. the funds are locked for a short period after.

spreading their stake over many different public key accounts in order to pay only small penalities for each violation. Finally, validators must reach agreement on the FBA channel before reactivating the PoS channel.

Quorum proofs of knowledge Finsense will use proofs-of-knowledge of quorum votes to reduce the communication complexity. Reducing communication size helps make synchrony more achievable. On the other hand, this must be done carefully so that the network is still able to detect accountable violations (i.e. when the adversary signs two conflicting messages).

5.3 Side-ledger interface staking

In addition to providing the infrastructure for public financial services, Findora can also connect to private ledgers. In this way, Findora can be seen as an Internet of independent (yet interoperable) financial networks, each with their own assets and financial agreements.

In the same way that accounts can issue assets on the Findora, *side-ledgers* (private and consortia ledgers) can set their own rules on asset issuance, transfer, approved identity providers, etc. Side-ledgers may also designate a smart account to serve as the gateway between the local ledger and Findora. This smart account, and associated nodes operating the account, are called a *side-ledger interface*. Essentially, side-ledgers seeking to connect to Findora for asset interoperability and liquidity are spokes of Findora's public infrastructure hub. Side-ledger interface accounts are required to stake FRA. And transactions transferring assets between the side-ledger and Findora incur a fee (also denominated in any combination of assets issued on Findora), as with any other transactions on Findora.

Assets issued on side-ledgers and Findora behave as follows when interoperating between the Findora and a side-ledger:

1. Assets issued on a side-ledger wishing to operate on Findora are simply abstracted as an asset issued by the side-ledger interface account on Findora. The side-ledger account must configure the issued asset to enable the required properties (e.g. free issuance of additional units)

2. Assets issued on Findora wishing to be operate on a side-ledger are simply sent to the side-ledger interface account. This account serves as the escrow account, should financial services on the side-ledger seek to validate the total supply of an asset on the side-ledger against Findora's ground truth. Side-ledger interface accounts may specify the asset

types, identity providers, and other arbitrary rules on transfers into the side-ledger interface accounts.

6 Application example: Smart Investment Funds

Smart Investment Funds (SIFs) are "smart contract" instantiations of the traditional investment fund; SIFs use the Findora developer tools to overcome the lack of transparency in investment funds today. Participants in a SIF include fund managers, investors, escrow agents, and target entities. Fund managers are responsible for configuring the initial structure of the fund and raising capital from investors. To achieve transparency, all capital is sent to the SIF over the distributed ledger in some type of digital currency, whether digital 'fiat' tokens backed by an issuing bank or a cryptocurrency. The assets are held in secure smart contracts until transparently delivered to the target entity/fund, whether a privately held company, a venture fund, or asset owner seeking capital.

In a typical private equity fund, investors first make capital commitments, assume the liability for uncalled capital, and deliver capital just-in-time when the GP makes a call for an investment. This is captured by making each capital commitment into a security token held by the fund manager. In a capital call, the capital commitment security tokens are exchanged for digital fiat tokens sent to the fund account, which the contract ensures are delivered to the target investee/debtor entities.

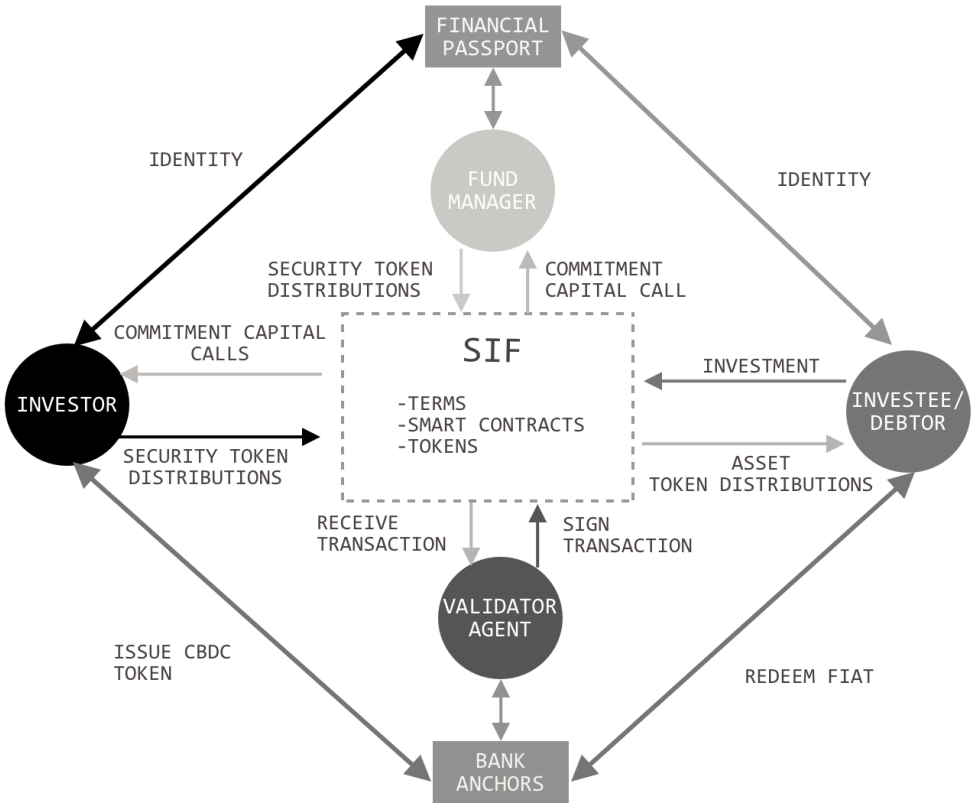


Figure 2: The interaction between investors, financial passports, investees/debtors, bank anchors, and validator agents centered around the SIF account. A SIF adds transparency into investment funds while using cryptography to preserve confidentiality, and enforces rules that can both prevent and expose fraud.

6.1 Fund managers

Fund managers issue legally binding digital tokens (security tokens) representing fractions of their funds to some or all of their investors. Fund managers still maintain the option to raise and operate part of their fund through traditional methods, and use of the SIF for part of the fund does not preclude or alter the process for the rest of the fund. Furthermore, tokenization enables fund managers to provide regulators and investors with

proofs of investment balance of record (IBOR) even if the overall holdings are confidential and hidden from the public. This capability is particularly germane to exchanges and money funds, which may hold a diversified class of securities backing investor deposits to maintain a stable Net Asset Value and guaranteed liquidity.

6.2 Validator agents

Validator agents are accounts that can operate as escrows or can enforce the rules of agreement terms between investors and the fund manager. For example, a validator agent can reject a transaction initiated by the fund manager that does not follow agreed-upon terms. The validator accounts can be controlled manually or can be tied to software that interprets rules in the SIF. The validators serve as co-signers on transactions between investors and fund managers. This is implemented through a *multi-signature account* on the underlying ledger protocol that is managed by the set of validator agents. Every transaction affecting this account requires the endorsement of each agent. This structure provides value above roles that are today handled non-transparently by fund managers, custodians, and administrators.

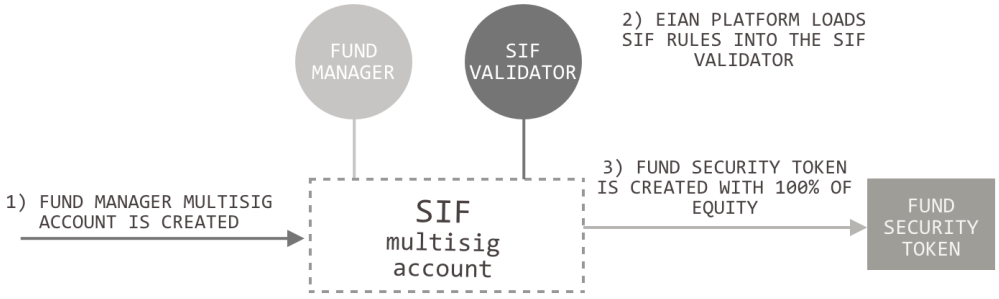


Figure 3: Illustration of a SIF multi-signature account for a fund with a Fund Manager and single SIF Validator. In general, there will be more than one Validator for a SIF account.

6.3 SIF rules

A SIF contains a set of rules that govern the behavior of the investors and the fund manager. SIF rules can restrict secondary market transfers, limit the list of allowed bank anchors, and specify fund manager fee withdrawals.

SIF rules can also extend to fund managers' holdings (e.g., target entities such as private companies, other funds, or asset holders).

As a simple example, Figure **??** below illustrates the flow for an investor depositing capital in a SIF that requires the investor to pass a KYC check. The investor issues a transaction to the SIF that includes a selective reveal of the investor's KYC information from their financial passport. In order to process this transaction the SIF must recognize one of the authorities who has signed the relevant information in the user's passport. A custom list of KYC authorities or root credential authorities could be programmed directly into the SIF during its setup, or the SIF may reference a dynamic list maintained by the underlying ledger protocol.

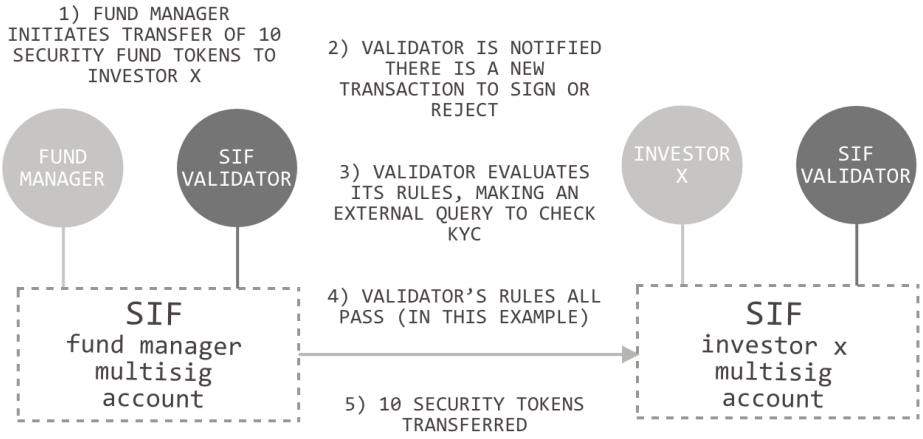


Figure 4: An investor deposits capital in a SIF that requires a KYC check.

6.4 Information visibility

SIFs limit the information visible to participants without relying on a single trusted party. They simultaneously achieve transparency and confidentiality through use of Findora's privacy-preserving compliance tools (Section 3.6).

The degree of confidentiality in a SIF is tunable, and we list several functionalities that we are able to achieve.

1. *Basic investor confidentiality (with fund manager visibility).*
   Hiding the amounts contributed by each investor to the fund from the

public (including the other investors) as well as the balances of each of the fund holdings can be achieved just from confidential payments. Only the fund manager is allowed to see all balances.

2. *Full investor confidentiality.*
Hiding from the fund manager the balances of each investor and revealing only the aggregate can be accomplished using Findora's confidential multi-source payment.

3. *Confidential aggregate balance.* In some cases it is desirable that only the fund manager knows how much has been invested overall. As noted earlier, Findora's confidential multi-source payment also supports hiding the aggregate amount from the public, and will also reveal to the fund manager whether or not the final sum it learns is correct.

4. *Conditional balance reveal.* A tweak to the above functionality would hide the balance of the fund from the manager unless its value surpasses a required threshold. This might be necessary for investor confidentiality when the investor pool is dynamic[43] across multiple attempts to fill the fund's round. This functionality will require more expensive privacy-preserving computation techniques.

7 Private investment markets

Private investment funds (venture, private equity, private real estate funds, etc..) are run by fund managers (i.e., general partners) who generally raise capital through established investor relationships. Investors trust fund managers to manage the fund according to rules and structure of the fund and faithfully employ their investment strategy.

Minimums to invest in private investment funds are high (usually $1M+) and given ten year fund lives and the illiquid nature of private investments, investors cannot expect liquidity events (sales or IPOs) for years. Moreover, awareness of and access to private funds is difficult for all but the wealthiest of individuals or families. This makes investing in private funds challenging and out of reach for most investors. On the flip side, most fund managers do not want money from individual investors because each investor takes as

[43]Consider a fund that recruits more investors until the total committed capital surpasses a threshold. If the incremental balance is revealed each time it is computed (even via MPC) then this could break investor confidentiality. In the extreme case there is only one new investor participating, then the difference in balances might reveal the individual new investor's balance (e.g. if the other parties did not change their investments).

much time to identify, market to, and manage as large institutions. For these reasons, despite being a top returning asset class, a very small percentage of accredited investors globally have commitments to private investments.

Addressing these issues is an ambitious endeavor involving a combination of financial and technical innovation. As a technical component, a Smart Investment Fund on Findora will help restructure investment vehicles to distribute trust, increase transparency, and improve the management of information, including transactions and credentials. This is only one part of a larger solution. *Ultimately, we are building a platform to increase access and make investing in private investment funds as safe and as simple as investing in public markets.*

7.1 The Findora Private Investment Service

Findora will provide FPIS (Findora Private Investment Service) as an early application built on top of the Findora Platform. It is a global, SaaS marketplace for investors and fund managers to more efficiently and equitably come together and transact. Each fund may use the service in addition to conventional methods of raising capital for all or a subset of their investors, allowing for an evolution of fundraising and operations rather than requiring a wholesale change to get started. Over time, we expect to see the creation of many other financial applications utilizing the platform.

The FPIS is designed to enable private fund managers to more easily raise capital, update limited partners and protect against internal fraud. The importance of protection against fraud is illustrated by what happened to Oak Investment Partners, a highly respected, thirty seven year old private equity fund. In 2015, it was revealed that a junior partner defrauded investors for \$65M[44]. This total breakdown of compliance did not happen in one transaction, but over nine events in a multi-year period. Suffice to say, if this breach can happen to a sophisticated, \$9B fund like Oak, it can happen to nearly any fund in the world.

7.1.1 Towards more inclusive markets

For accredited investors, FPIS provides the ability to invest in a top asset class, historically reserved for institutions and the super wealthy. Lower minimums and intermediate liquidity makes it more possible for the less affluent to participate, an important core value of our project. All offered

[44]https://thehustle.co/iftikar-ahmed-found-liable-for-defrauding-oak-partners/

funds will also be vetted (Know Your Fund, 'KYF') and required to follow processes to minimize the probability of fraud.

This functionality is consistent with the SEC's stated goal of wanting to enable 'Main Street' to benefit from the higher returns provided by private investments[45]. By having a platform that does the heavy lifting around KYC, AML, and compliance, regulators will be more able to do what they do best - put in place sound policy and ensure there are no bad actors. By designing the system such that bad actors are weeded out early, regulators around the globe can better realize the vision of inclusion.

7.1.2 Confidential proof of funds

FPIS will be able to operate a private equity fund whose balance sheet will remain private and confidential, but will have a publicly certified proof that its holdings are legitimate (i.e., from companies listed on a regulator-approved whitelist). The FPIS can also provide proof that returns to investors come directly from the fund's returns on its holdings and not from new investors. This reduces the risk of embezzlement, fraudulent investments in dummy corporations, or Ponzi schemes. Additionally, asset management events including capital calls, investments and management expenses can be validated. With this, people from every corner of the world can participate in a fair, transparent marketplace where information is trusted and confidential.

7.1.3 Intermediate liquidity

Most funds require multi-year "lock-ups" to match the "lifecycle" of private investing: from discovery, to company development, and finally through to harvest or liquidation of investments. This liquidity restriction is a significant barrier for many investors who would like the flexibility of selling their investments as needs arise. This is a cornerstone benefit of public market investing, and why investors accept lower returns for public fund investing versus private. The difference in return is known as the liquidity premium. The Findora Foundation's development of a secondary exchange and the use of security tokens to represent whole or fractional interests in an investment fund will help ameliorate this issue. As secondary markets increase liquidity, investors will be able to potentially trade their positions. For fund managers, a post-lockup trading market would provide liquidity for interested investors

[45]https://www.wsj.com/articles/sec-chairman-wants-to-let-more-main-street-investors-in-on-private-deals-1535648208

without requiring direct fund redemption or selling of assets, thereby protecting long-term holders. It also affords opportunities for smart investors to participate via secondary markets even after a fund has launched.

* * *