SKY PROTOCOL

DATA AVAILABILITY NETWORK

WHITEPAPER Version 2.1

This page is intentionally left blank.

Abstract

SKY Protocol is pioneering a Layer 2 solution tailored for Cardano, dedicated to revolutionizing data availability. SKY Protocol unbundles four essential aspects of blockchains—consensus, validation, data availability and bridging—so that each Decentralized Application (DApp) can use a combination of those aspects that best fits its needs. SKY Protocol's modular design enables better throughput, latency, privacy, safety, censorship-resistance and cost compared to existing blockchain technologies. There are of course tradeoffs involved among some of these qualities; but SKY Protocol enables each DApp to enjoy the qualities its users care about that existing one-size-fits-none systems can't provide, without having to pay a dear price for qualities they don't care about that these systems force upon them.

Future versions of SKY Protocol will support not only Cardano, but all blockchains that matter, and will include the ability for each DApp to directly bridge between blockchains; and they will also provide modular validation and consensus services as well as data availability. In the end, SKY Protocol aims at drastically reducing the cost and complexity of building DApps—so much that within a decade, the default would be for any new application to be decentralized rather than centralized. This abstract offers a comprehensive overview of SKY Protocol's architecture, functionalities, and transformative potential.

Disclaimer

The contents of this white paper are subject to potential revisions, may entail unforeseen risks, and could lead to new discoveries necessitating a reassessment of our initial assumptions. The SKY Protocol team retains the right to amend the white paper and project specifications for any reason. This white paper is intended to complement the code deployed by SKY Protocol. It's essential to underscore that the definitive source of accuracy and reliability is always the code itself. The technical information presented in this paper should not be construed as an exhaustive list of features, and it's worth noting that features may also be subject to removal.

Contents

1. Introduction

1.1. Summary

Today, Scalability—the ability to process a large number of transactions in a timely fashion—is a limiting factor for Decentralized Applications ("DApps"), to the point where most applications resort to a centralized operator. Most (all?) "Layer 2" solutions to scale the handling of decentralized assets from "Layer 1" blockchains sacrifice or compromise Decentralization one way or the other. The Cardano blockchain in particular, though promising by its robust design and healthy fundamentals, is lacking in scalability solutions, even centralized ones.

SKY Protocol is a modular solution to Layer 2 scalability that will launch this year (2024) on the Cardano blockchain, and on other blockchains afterwards. By going back to first principles, SKY Protocol can sport a modular design that pushes the boundaries of the tradeoffs involved; it thus allows each DApp to enjoy the capabilities and capacities it needs, without sacrificing decentralization, yet without burdening the necessary resource-limited Layer 1. Each DApp will have a DApp-appropriate level of throughput, latency to finality, smart contract capability, privacy, censorship-resistance, decentralization, cost-effectiveness, ability to interact with outside systems, etc. SKY Protocol will thus offer the Cardano community a rapid and secure DeFi experience.

Our choice of Cardano as a first blockchain to support is strategic for both SKY Protocol and Cardano: Cardano is unique both in (a) being designed for robustness and smart contracts, as well as (b) having a governance structure both decentralized yet able to deal with centralized institutions—thus providing the right environment for SKY Protocol to demonstrate its capabilities. Yet, Cardano also has many opportunities for improvements, and scalability in particular is a topic where it can best benefit from the contributions of SKY Protocol.

We'll explain these claims in more detail in this introductory section, then will give more details on the design of SKY Protocol.

1.2. The Problem: Scaling Decentralized Applications

1.2.1. Centralized Applications

Today, most software applications are centralized: they are run under the control of a central operator, whose code is secret, and who will spy on users, use ads to divert their attention, control what they are able to see, censor people or information they don't like, and otherwise serve the interests of the mighty to the detriment of the users.

Banks and credit card companies, even in so-called “Democratic First-World countries”, have been systematically refusing to process transactions for people who have the disfavor of the regime; not just sex workers, drug traders and gun dealers, but also journalists who will denounce corruption of officials instead of spreading their propaganda, truckers who peacefully protest their totalitarian lockdowns, or even friends and family of protestors who try to support them. Sometimes, the funds of the victims can even be locked or seized. Meanwhile, they are using their monopoly powers to conduct massive inflation, robbing the value of holdings of every productive citizen to transfer wealth to the pillars of the regime, as well as to fund massive wars, at unprecedented scale.

Social Networks and Search Engines, even in the same “Democratic First-World Countries”, have been massively censoring points of view that threaten the Establishment while pushing regime propaganda. Their so-called “fact checkers” are admittedly pushers of the official ideology.

The regimes often justify their massive spy networks in the name of saving the world from child abusers and terrorists, which is belied by numerous known cases where they do little to nothing against members of well-known child abuse rings (such as clients of Jeffrey Epstein), and have conducted massive funding of terrorist groups (such as Al Qaeda in Syria). But whatever the official justifications for central control, the actual result is that any centralized software application is soon captured by the Establishment to become part of a massive apparatus for the surveillance and brainwashing of the population.

Still, Centralized Applications (CApps) were historically discovered first, because they are simpler to build and run, and the problems with central control were secondary to having an application run at all to begin with.

1.2.2. Decentralized Applications

There is an alternative: Decentralized Applications (DApps), i.e. applications wherein no single participant or small group of participants holds the power (a) to control inflation to deal resources to themselves (“inflation resistance”), (b) to censor transactions they don’t like (“censorship resistance”), and sometimes even (c) to see the data of transactions of which they are not parties to determine whether they like them or not (“transaction privacy”).

The first successful decentralized application was Bitcoin, the peer-to-peer electronic cash system, that uses the “coins” managed by its ledger as incentives for people to honestly participate in the process of validating the “blocks” of transaction data that constitute the ledger itself, using the innovative “Proof of Work” mechanism to ensure consensus on “the” state of the ledger. Despite all its limitations and inefficiencies, Bitcoin proved that it was possible to create a monetary system free from central control, censorship or inflation, and remains a fast growing success to this day.

Since then, many Decentralized Applications have been designed; they usually reprise some variant of the "blockchain" data structure of Bitcoin (though some of them have neither blocks nor chains, but instead sequences of transactions or DAGs), or build additional structures on top of one that does. [TODO: give a non-exhaustive but diverse list of notable such systems—MasterCoin, Namecoin, Ethereum, Uniswap, xDAI, ENS, Solana, Optimism, Arbitrum, Celestia, etc.]

These systems hold the promise of a world where users are not held hostage by their software service providers anymore.

However, this promise is still largely unfulfilled to this day. DApps do help a lot of people evade the restrictions from the centralized monopolies in the more oppressive countries. Yet overall, most blockchains seem to be mostly vehicles for speculation, and not much in terms of actual platforms for DApps.

1.2.3. Technical Hurdles to Decentralization

Today, writing a DApp is extremely difficult. Running one is expensive. And existing systems can only handle so much transaction volume.

For instance, as of early June 2024, the Bitcoin network can handle about 2800 simple transactions every block (1MB limit, about 374 vB per simple transaction), one block every 10 minutes, for an average peak throughput of about 4.6 transactions per second ("tps"), with a cost of about $0.81 to get a first confirmation within 60 minutes for a typical simple transaction. Its rival, Ethereum can handle about 714 transactions (15M gas limit, 21000 minimum gas per transaction) per block, one block every 13s, for an average peak throughput of about 55 tps, at a cost of about $0.73 to get a first confirmation within 3 minutes for a typical simple transaction. Neither the throughput limits nor the transaction costs make either system acceptable as a replacement for centralized systems, that can each typically handle many tens of thousands of transactions per second, and beyond.

Some newer blockchains, such as Solana, TON or Tezos, have recently advertised performance that rivals with centralized credit card processing systems. This is great progress, but even then, this is still far from what is needed to move all human software to DApps; and then there is the problem of bridging those systems with those other systems, where the valuable tokens are, or where the useful action happens.·

1.2.4. Economic Limitations to Solving Decentralization Issues

Many DApps seek to address scalability issues with blockchain technology.

Historically, DApps were first built by defining protocols that were each specific to a single DApp. Then they needed to raise capital around a new token for a new economic validation

network, from scratch, but the capital raised will only be used to secure that single DApp, which is not efficient.

Ethereum and other blockchains with "smart contracts" (Cardano, Solana, Tezos, Nervos; Cosmos, PolkaDot, etc.) have each tried to offer a universal platform on top to write DApps with (or sometimes without) a Virtual Machine. But apart from the many other issues they may have had, these "Layer 1" blockchains have all had issues with scaling—and though there has been progress in this regard, the situation is still very far from one when all CApps could be converted to DApps—there is just not enough throughput, and what throughput there is is far too expensive for most applications. There again, existing systems are not efficient enough.

1.2.5. Data Availability and its Existing Engines

One promising venue to enable massive scaling wherein DApps can compete economically with CApps is specialized Data Availability (DA) "engines"—subsystems that specialize in one single fundamental aspect of blockchains wherein they achieve (or fail to achieve) scalability.

Several projects have tried recently to provide this service. The first was Celestia, and many followed. We will call them all "first generation" Data Availability engines. It is interesting to study their successes and failures (so far):

- Celestia is historically the first project that tried to explicitly solve the Data Availability problem. What more it does this without trying to push a specific consensus layer onto its users. It therefore offers a more modular design than PolkaDot or Cosmos, that do try to have you adopt their consensus. But its threat model has a few flaws (see next section).
- Avail tries to build a Celestia-like system that will further bridge all the blockchains.
- EigenDA has a few good ideas, especially for how a token can be truly universal; however it doesn't address some of the flaws of , and we have to see what really comes out of it.
- NearDA… ? [TODO dig deeper on competition]

Note that "rollups", the use of Layer 1 blockchains themselves as Data Availability engines for side-chains that use them, was invented as a clever interim solution to improve scalability a little bit until either Layer 1s learn to scale properly, or a good well-trusted DA engine makes scalability happen.

1.2.5. The Absurdity of "Data Availability Sampling"

Many first-generation Data Availability networks feature some client-side "data availability sampling" that they claim contributes to security by detecting when network participants are dishonest. However, the technique fails to actually improve security:

1. The sampling comes too late to save any user's assets.
2. The sampling provides no means to sanction the dishonest data availability node.
3. When unavailability is detected, the sampling cannot distinguish between bad data availability node and bad side-chain operator.
4. Dishonest data availability nodes can distinguish between usage by actual side-chain watchers and mere random checks, and only provide uselessly limited data, slowly, to those who don't need it.
5. In its "efficient" version, this sampling does not at all check availability of the data, only a concise zero-knowledge proof of possession of the data. However, lack of possession of the data was never the problem to begin with—in a data withholding attack, the Adversary is a dishonest Side-chain Operator (SO), who does possess the data, but is precisely withholding that data and keeping the assets hostage. At that point, the proof of knowledge is just like a hostage taker sending an ear of the hostage to prove he was still alive recently—the data wasn't released at all, and you still don't have it, but now you know the hostage taker means business.

Client-side Data Availability Sampling is therefore worse than worthless: it comes with a lot of complexity to yield a false sense of security, and actually makes the job easier for hostage takers.

1.3. The Solution: SKY Protocol

1.3.1 The Challenge

Now what if we could radically improve DApp technology, so it becomes easier and cheaper to develop DApps than Centralized Applications (CApps)? What if DApps could be made to scale just as well as CApps, at a running cost lower than that of CApps, while providing better guarantees for the end-users? We argue this is possible, but requires going back to first principles to understand how DApps and CApps work.

1.3.2. Value Proposition

SKY Protocol proposes a modular architecture for building DApps that:

1. Radically simplifies the work required to build DApps.
2. Allows DApps to seamlessly scale as their usage grows.
3. Will eventually support bridging to all blockchains.

The end-goal is that eventually, most people writing new Apps in the future will write DApps by default, not CApps, and most DApps will use SKY Protocol.

SKY Protocol will simplify the work required to build DApps by offering a modular architecture (see section 2 below), wherein developers can mix-and-match components for consensus, validation, availability, and bridging. Moreover, SKY Protocol will enable developers to metaprogram all the parts of the DApp (on-chain for every chain at stake plus off-chain for every participant) from a single high-level specification, whereas today you have to manually and redundantly program those many parts and ensure they are and remain in synch and without exploitable bug as the project evolves.

SKY Protocol will allow DApps to seamlessly scale by automatically adapting validator supply to validation demand: The more people want to use SKY Protocol's network, the more the network splits into shards, keeping the overall cost and, at equilibrium, the price, about the same for each megabyte published and kept available for a day. As the network grows, though, the total fee market grows proportionally with the usage of the network, and thus the SKY Protocol token used to split the work and profits of partaking in the network itself grows in value.

SKY Protocol will eventually support bridges to all blockchains. Obviously there are many potential security issues with bridging; each time the bridging functionality is extended, the change will therefore require a community consensus as well as extensive testing (and potentially use of formal methods to qualify the code).

1.3.3. Choosing Cardano

SKY Protocol has the ambition to eventually support all blockchains. Still, SKY Protocol must start with some blockchain, and be careful where to start. The first blockchain should have:

1. A robust consensus algorithm, so SKY Protocol can focus on data availability.
2. Support for smart contract validation, so SKY Protocol can focus on data availability.
3. Issued assets, so SKY Protocol can host its native asset on that blockchain.
4. At the same time, a relative lack of existing scalability solutions on the target blockchain means Sky will corner the market of providing an essential service to that community.

These are many reasons why SKY Protocol and Cardano are a great match:

1. Cardano provides a very robust consensus algorithm, based on much academic research, as well as proven in practice by years of deployment in production.
2. Cardano supports smart contracts that are much more powerful than exists on other UTXO-based blockchains, while its data model is based on UTXO rather than Accounts which makes for easier, parallelizable validation of past transactions, which matters for scaling (see our AVOUM whitepaper).
3. Cardano has a builtin notion of issued assets so there is much less aggravation and security risk in using those assets than from e.g. an Ethereum contract.

4. Cardano doesn't yet have a good Layer 2 ecosystem, and even less so a Data Availability Engine, so there is a lot of potential synergy in becoming the first such solution for Cardano.

There is thus a synergy between SKY Protocol and Cardano, both in the short run and the long run: in the short run, SKY Protocol fulfills an immediate market need for Cardano—Scalability via a Data Availability service—while Cardano provides a niche market for SKY Protocol—a welcoming community eager to use the service. In the long run, shared technical choices make for a good partnership: high-level languages, UTXO, powerful contracts, a builtin notion of issued assets, etc.

1.3.4. SKY Protocol Summary

SKY Protocol will construct a Data Availability network tailored for use on Cardano. Many Cardano-based DApps will be able to leverage our network at the same time, thus pooling resources and capital for network validation.

Our network will ensure side-chain data is available, and generate attestations in a format that allows seamless use by Cardano smart contracts. Our network will enable Cardano Side-chains that take custody of tokens defined on Cardano and *securely* manage transactions at a scale far beyond what is possible with Layer 1 solutions. We will support DApp validation through both "optimistic" interactive proofs and zero-knowledge non-interactive proofs.

Our Data Availability Engine will itself be secured by a Proof-of-Stake economic validation network, for which we will launch a SKY staking token on Cardano: data space will be divided into independent topics (to provide indefinite scaling), and topics into redundant shards (to provide resistance to failures and attacks); time will be divided into slots (to account for varying consensus); for each slot, a committee will be determined in advance in a deterministic pseudo-random manner, based on stakes at the time of determination (to provide economic security); and another committee will watch the committee of the current time slot to ensure honest behavior (to keep nodes honest and the network healthy).

Additionally, our network will progressively introduce censorship-resistant features, including time-locked blinding of data to prevent selective censorship by committee members, and onion routing of requests to prevent client discrimination. We will also introduce lower-latency transactions via a user compensation out of operator collateral in case of operator failure or fraud.

This comprehensive approach empowers DApp developers to craft high-throughput, low-latency applications or highly censorship-resistant, attack-resistant applications, all with reduced costs, anchored on the Cardano blockchain.

2. Concepts and Overview

2.1. Decomposing Blockchains

2.1.1. Modular DApps

In our Legicash Whitepaper in 2018, already, we proposed to scale blockchains in a decentralized way, by creating Layer 2 side-chains supported by interactive validation of data posted onto a registry for shared knowledge—techniques now known as "optimistic validation" and "data availability engine", that fill the holes in the previous Plasma Whitepaper. We later named our next (and still current) startup "Mutual Knowledge Systems" in reference to mutual knowledge, the term in epistemic logic and game theory for knowledge shared between all participants in a system—which is what our proposed registry creates.

Since then, systems following this design have been created. Celestia notably implemented this design in a modular way—though their proposed decomposition into execution, settlement, consensus and data availability is slightly problematic, and we propose a better conceptual decomposition below.

2.1.2. Consensus

The best-known service provided by a blockchain (or related architecture) is *consensus*: the ability to have all participants agree in bounded time on what is *the* evolving current state of the system at any moment (or at least, what it was quite recently).

The invention of the Nakamoto Consensus via Proof-of-Work was the innovation by Bitcoin that crucially enabled fully Decentralized Applications. Since then other solutions with different tradeoffs have been discovered, most notably Proof-of-Stake and its many variants.

2.1.3. Validation

We can distinguish from the service of consensus strictly speaking the service of *validation* of the transactions (atomic state changes) that the system executes from one agreed upon state to the next.

The Celestia authors may call this service an "execution" layer, but in general the blockchain need not actually execute anything: for instance, when using zero-knowledge proofs ("zk-proofs"), no transaction execution takes place on the blockchain as such; instead, transaction execution happens outside of the blockchain, privately within clients, that generate proofs; and all the blockchain servers do is check that the proofs are valid, without even having to know what kind of transactions the proofs are about, or how much of what assets they are

about. Also, this service is not a “layer” in any meaningful way: there is no way to say that one is above the other or any such thing.

Ways to validate a transaction include (1) indeed having the servers execute every step of every transaction, as in Ethereum, or (2) letting users issue claims, and using an “interactive proof” to establish which of several conflicting claims shall be rewarded or punished, as in the Legicash Whitepaper, and as Optimism and Arbitrum promised to do but never delivered on, or (3) requiring users to publish “non-interactive proofs” using e.g. zk-SNARKs, which can be seen as paying in advance the worst-case price of interactive proofs, in exchange for which reducing the cost and latency of validating transactions.

2.1.4. Data Availability

One last service necessary to complete a Layer 1 blockchains is *data availability*: all participants must see all data so as to be able to validate the data, join the network, see which transactions did or didn’t make it, assess the current state of accounts, and generate new transactions.

Layer 1 blockchains implicitly provide and enforce Data Availability, without the issue being usually dwelled upon in whitepapers and documentation: miners who partake in the consensus just refuse to accept, validate and build upon blocks of which they haven’t seen the complete contents, and those blocks therefore are never included in the chain. Miners who withhold the contents of their blocks therefore forfeit any fees and reward from those blocks as well as the resources to mine them.

Yet the question of Data Availability becomes an issue to explicitly address when discussing Layer 2 solutions: the Plasma whitepaper (2017) famously mentioned *data withholding attacks* as an issue that could allow failing operators to cause a side-chain to become unavailable, or, depending on the design of the side-chain, even allow dishonest chain operators to abscond with their users’ assets.

A solution to this issue is usually called a Data Availability Engine. A popular solution on Ethereum is to publish the data for the Layer 2 on the Layer 1 itself, in a process that Vitalik Buterin calls *rollup*, and that Ethereum now specially supports with its *Proto-Danksharding* (EIP-4844). However, this solution only works if the Layer 1 blockchain is Ethereum (or *mutatis mutandis* if the Layer 1 on top of which a Layer 2 is built supports an affordable similar mechanism); what more it remains somewhat expensive due to the high cost of gas on Ethereum in general (and presumably on any Layer 1 that would become successful enough for a Layer 2 atop it to be useful).

Data Availability is the service that the SKY Protocol will specifically provide through its network of validators.

2.1.5. Bridging

A fourth and last service that a DApp may optionally provide is *bridging* with other systems: communication with these other systems (input and output), actions on these other systems, etc.

Note that this service is *optional*: Bitcoin, and most "Layer 1" blockchains after it, do not usually provide any such service. Transactions processed by the system can only affect or be affected by entities *within* the system, never entities *outside* it. On the other hand, many "Layer 2" solutions do provide bridging:

- Bridges, after which the name "bridging" was chosen, are DApps that connect two or more other Decentralized Systems (blockchains).
- Oracles are DApps that connect Decentralized Systems with Centralized Systems outside the world of blockchains.
- Side-chains are Decentralized Systems with a builtin bridge to a "main chain", typically a popular Layer 1 blockchain.
- Payment Channels and State Channels can be seen as special cases of Side-chains as above, between a small fixed committee of participants using unanimity as consensus.

The Celestia documentation talks about this fourth service as "settlement", but that is a bad way to look at it: First, because bridging can be used for much more than settlement, as illustrated by Oracles. Second, because blockchains, whether "Layer 1" or "Layer 2" can settle transactions in their native assets without further service beyond consensus, validation and data availability. Third, because there is no reason why a "Layer 1" blockchain couldn't directly handle bridging with outside systems as part of its own protocol.

2.1.6. Composing Interchangeable Services

The four services above are largely independent, and it is possible to mix-and-match between several versions of these services as you develop DApps. These services can themselves be parameterized, or layered on top of each other (as in a Side-chain on top of a Main-chain), etc. In the end, rather than a one-size-fits-all blockchain, we can offer a custom kit to build DApps in a modular, composable way.

Other projects have tried this before, notably Cosmos and PolkaDot. But they fall short of the ideal in many ways.

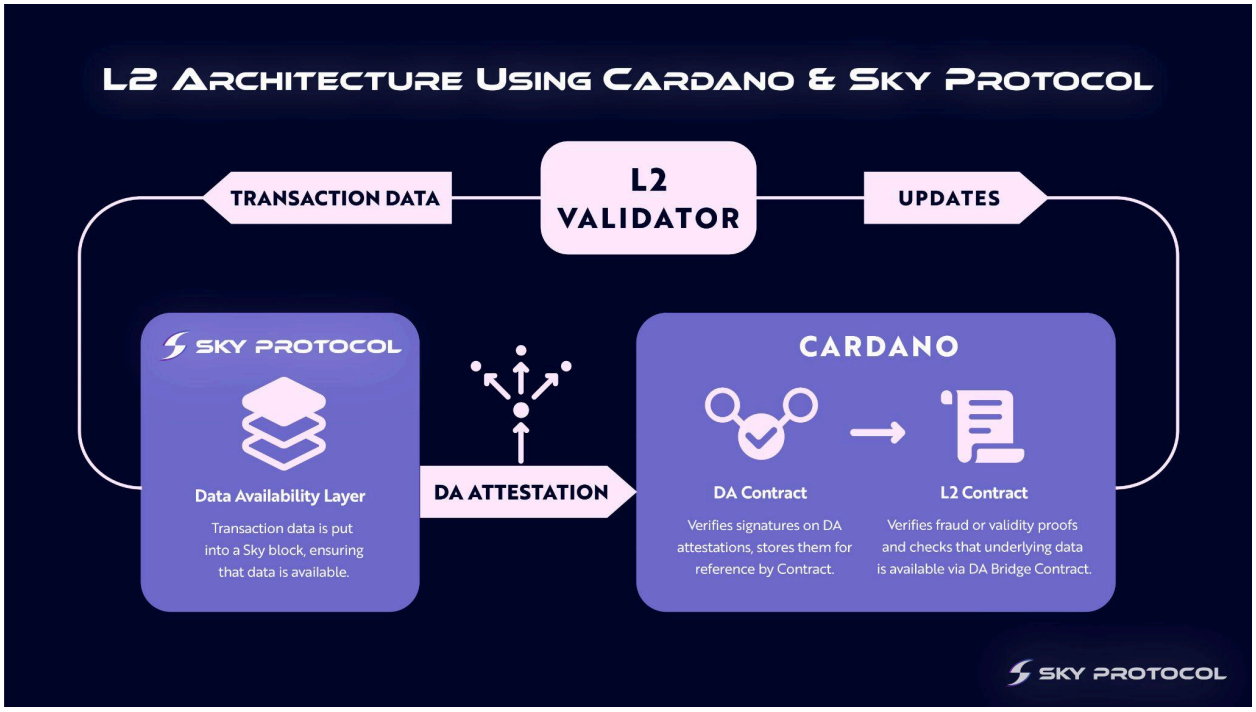
First, they use relatively low-level languages that are very well suited for the development of a networked server, but not for the co-development of matching servers and clients and operators and other participants in many roles with matching code on-chain and off-chain, and if there is any bug you lose your money. For this kind of challenge, you will want more of a functional programming language with suitable metaprogramming, so that all the many parts of a DApp can be automatically generated from a single specification. Our team has experience doing this

with our language Glow, a language that can generate both smart contract and matching client code from a single specification: we prototyped in Scheme and reimplemented in Agda. See section about *Glow* below.

Second, these ecosystems are largely incompatible with other Decentralized Systems, and are mostly gimmicks to sell more of the Cosmos token ATOM, and the PolkaDot token DOT, respectively. SKY Protocol is designed to eventually work with all existing blockchains rather than requiring users to use SKY Protocol's consensus and associated tokens. SKY Protocol supports and encourages the use of other blockchains and their consensus, either as "the" consensus provider for your DApp, or as chains with which your DApp is bridged. While SKY Protocol will have a consensus of its own to foster the internal security of its own protocol, and that consensus can indeed be used by DApps, it is not the product being advertised, sold and optimized for: the Data Availability service is.

Third, in the particular case of Cosmos, the economic validation is even worse, because each DApp needs its own committee, the capital of which must be raised from scratch as a tiny fraction of the capital already raised by Cosmos. The Cosmos situation was made necessary by Cosmos having failed to define a secure metered VM, and so not being able to securely and efficiently share work between its "app-chains". By contrast, Polkadot having embraced WASM as its secure metered VM can have its consensus validate updates from its "parachains". SKY Protocol will let every DApp developer choose the approach they prefer, but will support and encourage models closer to Polkadot than to Cosmos in this regard.

2.2. SKY Protocol Architecture



2.2.1. Modularity and Options

There will be many ways to build applications with SKY Protocol, in a modular way that commoditizes the many components of a DApp: consensus, validation, data availability and bridging. Developers and eventually users will be able to pick whichever component makes most sense for them for each instance of each DApp, sometimes dynamically, given considerations such as security and cost effectiveness.

Now, we will start our work with a component that has the lowest hanging fruit for DApps today: Data Availability. While there are plenty of relatively mature platforms for consensus and validation, and somewhat acceptable solutions for bridging, data availability is still largely unsolved, though competitors are already lining up on the topic.

2.2.2. Data Availability Engine

The first component we are building is a pure Data Availability (DA) Engine. This DA can help keep side-chain operators honest and build trust in a decentralized side-chain solution: a side-chain anchoring smart contract forces operators to publish all their data on the DA; then users know they and their agents too can validate the data and build models sufficient to keep issuing side-chain transactions and main-chain transactions to secure the assets. Thanks to their accurate model of the side-chain, they can ensure that they can get their assets out of the

side-chain in case of operator failure, whereas no dishonest person will be able to take the assets away.

Furthermore, validation of side-chain transactions is either done using interactive proofs ("optimism") or non-interactive proofs (zero-knowledge). In the case of interactive proofs, there must be at least one honest watcher at any given time reading and validating all the transactions fast enough, and latency is added after the transactions are initially published to leave watchers time to challenge them. In the case of non-interactive proofs, someone must be incentivized to write the proof in advance of publishing the transactions, which adds latency before the transactions are initially published.

Economically, a Data Availability engine enables many DApps to share a common economic validation network for the crucial aspect of Data Availability, just like they may share a common economic validation network for the crucial aspects of Consensus and Validation, from the Layer 1 main chain of which they are a side-chain. By sharing a network, these DApps effectively pool resources together to achieve greater security against economic attacks than possible if operating apart.

Note that the DA engine itself will itself need a Consensus aspect, and a Validation aspect, and that Layer 2 DApps could then reuse these aspects, at which point they are "just" smart contracts running on the DA seen itself as a Layer 1, and they require a single economic validation network. You could also see every DApp that uses a different Layer 1 for its consensus as being a bridge between the DA network and that Layer 1 network.

2.2.3. Mutual Knowledge vs Common Knowledge

Mathematicians who study Epistemic Logic or Game Theory distinguish two concepts, Mutual Knowledge and Common Knowledge, that are very relevant in the context of DApps: Mutual Knowledge is what every participant in some system know; whereas Common Knowledge is what every participant knows that every participant knows that every participant knows, etc., recursively.

For instance, consider an international conference where everyone speaks English by default. And at that conference, consider a group of people talking to each other. If every person in the group knows and prefers to speak Spanish, then knowledge of Spanish is Mutual Knowledge; but they will keep speaking English because they don't know that they all speak and prefer Spanish. If someone realizes that this might be the case, and asks aloud if anyone in the group doesn't prefer Spanish, and no one chimes in, then knowledge of Spanish becomes Common Knowledge, and everyone will start to speak Spanish.

Mutual Knowledge is independent knowledge by each participant, and can be achieved in parallel, without interaction between these participants. Common Knowledge on the other hand

includes Mutual Knowledge and beyond it some *agreement* about the facts at stake; it requires some common public interaction between the participants, some kind of simultaneous handshake of all, or unchallenged reliable broadcast—something notably *more* than asynchronous peer-to-peer communication: at least something like a real time clock and assurances

A humorous illustration of the different between Mutual Knowledge and Common Knowledge is provided in *Friends* S5E14 "The One Where Everybody Finds Out", in which all protagonists know of a relationship, but behave in funny ways because they do not know that the other ones also know—or, later, some know that the others know, but the latter don't know that the former know.

The concepts are relevant to DApps, because the *Consensus is a variant of Common Knowledge*, whereas *Data Availability is a variant of Mutual Knowledge*. Indeed, the Consensus of a blockchain or DApp is a form of Common Knowledge, wherein participants holding the majority of mining power all agree on updates to the system. And Data Availability is a form of Mutual Knowledge, wherein participants holding the majority of mining power all have independent knowledge of some data, without there being any agreement about it (yet).

We can thus use the well-known formal tools of epistemic logic to model the behavior of blockchains, where information goes from Knowledge propagated through some Gossip Protocol until it becomes Mutual Knowledge in the Data Availability layer that later becomes Common Knowledge through Consensus. Finally, the Common Knowledge decides of the geometry of the network (who are the members with how much stake, and who among them will play what role(s) in the network at the next time slot), in a feedback loop from the Consensus back to the Gossip and Data Availability.

We also understand why Data Availability is a necessary step toward the establishment of Consensus, and therefore why the latter is necessarily more expensive to achieve than the former. Not only that, we understand why Data Availability can be achieved in a massively parallel way with little interaction between participants, while Consensus is an intrinsically sequential process that involves a lot of interactions between all the participants—and some mechanism beyond simple peer-to-peer communication, such as a reference clock. Finally, we understand that Data Availability is often enough to validate transactions—when there is no conflict—whereas Consensus is ultimately required to validate transactions in presence of conflicts, by choosing a priority ordering between transactions that resolves the conflict.

The insight of Data Availability as Mutual Knowledge and how it can be used to scale DApps is the reason why we named our company Mutual Knowledge Systems back in 2019.

2.2.4. Limits to Capacity

Since Mutual Knowledge as such requires no interaction between participants, it can be readily scaled "horizontally" via massive parallelization. There are however some limits to this scaling:

1. Total network capacity: Our stakers can always reserve more machines to match the demand, as long as there is demand for data availability at a price higher than the operating costs. Therefore, the network will reach an economic limit before it reaches the technical limit, which is the capacity of the entire Internet.
2. Activity factor: The effective capacity of the network is *divided* by an activity factor, the average number of Data Availability participants (publishers, validators, watchers) for any given piece of data (that will divide the capacity). We should typically assume there are a few hundreds (or thousands?) of participants interested in a topic at a given time.
3. Security factor: The effective capacity of the network is further *divided* by a security factor that accounts for the redundancy in copies and variants of the data required to keep the network robust. Our design has a security factor of 3.
4. Consensus latency: The Data Availability network needs to reach a Consensus on the metadata (if not the data), which generates additional traffic and has additional latency of its own (though it does not slow down Data Availability latency as such). This multiplies latency to Consensus by the base two logarithm of the number of members in each committee, multiplied by the number of nested consensus layers, multiplied by some small security factor.
5. The capacity cannot be discovered in a fully automatic way, or the discovery algorithm could be abused by attackers so as to grow or shrink too fast, breaking the network. Instead, the members of the network must watch it and vote for or against capacity changes, though change proposals can be automated.
6. For the sake of decentralization, the load capacity for each individual Data Availability network participant must be small enough that a relatively small machine can partake.

Still, even within these limitations, Data Availability can scale DApps much further than possible with Layer 1 solutions.

2.2.5. Topics and Shards

SKY Protocol achieves parallelism by dividing its network into mostly independent *topics*. Each topic is being served by its own *topic committee* that accepts data, makes it available to watchers, and publishes attestations to this effect. For the sake of attack-resistance (which subsumes fault-tolerance), each topic is further divided into *shards*, each served by a single committee member, such that only a fraction of the shards is enough to reconstitute the data being made available: at least 67% of the committee must sign an attestation of availability for it to be considered valid, and at least 34% of the committee must actually serve the data for the data to be available. Finally, time is divided into *slots* of half a second during which the

committee is tasked with accepting new data for the topic, and in publishing periods of a week during which the data remains available.

Each topic achieves consensus about what data was served, what fees were collected and earned by whom, which committee members misbehaved (if any), and any other relevant metadata. Consensus about what happened during a time slot will take the duration of several time slots before it converges.

Each topic has relatively small limits, which guarantees that watchers can't be flooded by attackers while listening to the topic (see the Threat Model below). To scale, a DApp may require its watchers to watch as many distinct topics as necessary to fit its expected usage load. Changing the number of topics and specific list of topics for a DApp may involve consensus updates on both the main chain that the DApp is hanging to, and the Data Availability layer.

2.2.6. Autoscaling

As the total use of the Data Availability platform grows, more topics need to be added. If some topics are empty while others are full, we will want to somehow group together topics that are not full under a single committee, with shared limits between them. Conversely, when a topic is subject to a lot of contention between DApps (or with spam), one solution is for a topic to provide a reserved bandwidth for a single DApp. At least in a first time, the adaptation will be done outside the DA itself, by the applications using it: those with low throughput for the DA (but high for the Layer 1) may want to share a topic with no authentication (or shared authentication) with other applications. In the future, topics could be enriched with more and more general validation rules beside checking a signature.

Existing systems do that by requiring DA node operators to collectively possess economic expertise and vote every so often to set the right side for the pool.

But the real innovation will be for topic reservation to be a double auction with the Sky Protocol as a single collective seller (for security reasons, based on median price or some such?), and block publishers as buyers. As contrasted with first-generation Data Availability networks, Sky Protocol will be a DEX for Block Space.

Now, any change in the consensus with regards to topics—how many there are, how they are grouped or reserved or validated in any way, is subject to voting by humans, because any fixed and known auto-scaling algorithm can be gamed.

2.2.7. Attestations

A side-chain operator (SO) using SKY Protocol as a Data Availability engine may get a separate attestation from each Data Availability committee member for the topic (DA) immediately after posting data to said committee member. He may then organize all those attestations into large proof of availability, that he can post as is, or further summarize using zero-knowledge proof.

Later on, as the topic committee reaches consensus, and the general consensus includes the report from the topic committee, shorter and simpler proofs of availability are possible.

Depending on their latency requirements, smart contracts may recognize all these kinds of attestations, or only the shorter and simpler ones that only become available later but are cheaper to check in a smart contract.

2.2.8. Layered DA Architecture

We anticipate that it is largely possible to write the DA in layers, such that the system is useful at each level of provided functionality, that only depends on previous layers:

1. Protocol for individual DAs to each independently accept data and republish it, independently from other DAs.
2. A user-side aggregation protocol given a committee of individual DAs (e.g. from Proof-of-Authority, a.k.a. PoA) can create a synthetic DA that can resist ⅓ attacks.
3. A protocol to generate shorter synthetic signatures from the committee, that can be used by our bridge to Cardano (and, later, other blockchains).

At that point, we have a working DA, albeit without the decentralization and accounting that keeps interests aligned, yet. A second phase will implement them as follows:

4. A consensus that can manage a ledger of tokens, for the moment based on the existing committee and without a smart contract virtual machine.
5. A payment protocol for users to independently pay each DA for its service.
6. Staking, wherein token holders stake tokens and delegate mining power to miners, and the consensus can reward or slash miners and stakers based on miner behavior as observed by a ⅔+ε majority of the consensus.
7. Proof-of-Stake decentralization (a.k.a. PoS), wherein the committee is not fixed, but determined randomly at each epoch based on a Verifiable Random Function (a.k.a. VRF) and the previous distribution of stakes.

A third phase implements scaling via the “DEX for block space” idea:

8. A price auction for the committee to agree on a price at which to sell block space to users.
9. Many parallel consensuses, each with its own committee that drives a topic, based on the same Proof-of-Stake, to scale further while maintaining small node size.
10. A hierarchical consensus that synchronizes the above parallel consensuses together. (Note however that applying the security construct recursively cannot strengthen security: each iteration increases costs and latency, and the attackers can organize and focus their resources on the weakest link they can dynamically detect or create, even though the average case might be stronger.)

11. Dynamic resizing of the number of subconsensuses based on matching the demand with the supply from the auction of the sixth layer.

The above yields a complete production version of Sky Protocol. Further layers are possible:

- A state channel payment network to take some load off the chains as users pay for service, which can be done faster and at a finer granularity, too.
- A virtual machine for more general smart contracts to enable secondary applications, bridges, etc., that can leverage the DA network and bring new functionality. These smart contracts could use the kind of safe cross-shard communication strategies we built for Harmony One (Whitepaper, Gitlab).
- An integration of validation services together with our DA services, so our blockchain can itself host side-chains using its own DA services. Integrate both interactive ("optimistic") and non-interactive (zkproof) validation services.
- An integration between our short-term DA network and medium or long-term storage networks.
- Microaccounting to properly reward low-stake nodes that participate in the gossip network and help the network scale.
- Mechanisms to segreggate topics by security levels, so safer, higher security topics are not run on the same servers as lower security topics.
- An onion routing or mix network layer to make censorship harder.
- A more declarative and portable language for building DApps (see our Glow language).
- etc.

In the end, SKY Protocol can become a complete L1 network with many services beyond those in existing networks.

2.2.9. Censorship-resistant Gossip

In a future variant of SKY Protocol, we will address the possibility of censorship at the level of the gossip network.

One technique we will use is some form of onion routing or mix network, with the routing path again randomly selected based on stake, so as to make it hard to impossible for nodes to behave differently based on who they are talking to—they have to either serve everyone the same, or provide service to no one at all, but they cannot discriminate and fake making the data available when watched by committee members yet not actually make it available when queried by regular users. Any such technique will add a lot of latency while cutting down throughput, and should only be used in an opt-in per-topic way. However, the "toplevel" topic might have to enable it and pay this price, so the network as a whole can remain censorship-resistant. For the sake of performance, most regular traffic could and should remain out of that toplevel topic.

2.3. Threat Model for Data Availability

The threat model is the most important feature of a decentralized system. Indeed, decentralized systems are open to interaction with anyone, including dishonest people. Without a proper threat model, determining what attacks are possible and having appropriate defenses, the honest users are left helpless at the mercy of dishonest users, and any utility the system provides becomes an attractive nuisance wherein honest fools are only attracted later to be robbed by the dishonest ones.

The specific design for our network, as for any decentralized system, is largely based on ensuring there are proper counter-measures to the many anticipated attack scenarios.

Our threat model includes potential bad behavior from Data Availability Nodes (DA), Side-chain Operators (SO), Side-chain Watchers (SW), Side-chain Users (SU), who could, sometimes in collusion with each other, try the following attacks, for which we have prepared the following appropriate defenses.

2.3.1. SU Double-spend

Issue: SU tries to double-spend alone.

Solution: Caught by an honest SO. No DA needed for that. DA protects against dishonest SO.

2.3.2. SO Double-spend

Issue: SO tries to double-spend with assumed SU identity.

Solution: Caught by honest SW if DA is used and working; caught by super-majority honest DA committee if SO tries not to use DA.

2.3.3. SW sleeping on the job

Issue: No one pays SW, or SW is dishonest and does not do its job. There is a coordination problem for SUs who need to collectively ensure that at least one honest SW is running at all times (and resist other attacks such as DDoS, legal attacks, power failures, etc.), though none of them would like to pay the full price of it.

Solution: This issue is usually solved not through technical means as such, but through social norms: side-chain users are expected to semi-randomly subsidize independent reputable SWs, at least only one of which will stay honest. Alternatively, a side-chain may use slower and more expensive non-interactive proofs (i.e. zkSNARKs) instead of interacting proofs, which moves the issue to only one of liveness rather than safety. Eventually, more of the validation can be eventually moved to the DA network (until then it’s not an issue with the DA strictly speaking, though still an issue with modular designs that use DAs).

2.3.4. SO Overload

Issue: SO generates a large volume of transactions until all honest SWs have to give up, at which point they can insert double-spending transactions that no one can verify is bad (if using interactive proofs), and/or that no one else can keep building on (if using non-interactive proofs)

Solution: Have the DA enforce throughput limits for each side-chain and/or “topic”.

Note on competition: We are not aware of other DA solutions having defenses against this class of attacks. And yet, it is a necessary feature of a network that enables more scalability than any single client can handle.

2.3.5. DA complicit in Data Withholding or Overload

Issue: Dishonest DA nodes help dishonest SO in trying to prevent SW from doing their job.

Solution: Have other DAs watch each other, with slashing of stake from dishonest ones. This requires the network to produce or at least use actual consensus, not just Data Availability (though Data Availability can come first, and turn into consensus later; or the two can remain separate).

Validators found not to do their job properly will see their stake slashed. Watchers from the SKY Protocol will have the same behavior as watchers for side-chains as data availability clients, so that nodes cannot use different behavior to discriminate between clients. Data will be divided between many shards so that neither node failure nor dishonesty can cause unavailability, within the usual model of 67% honest nodes.

2.3.6. DA collecting fees without working

Issue: Dishonest or otherwise broken DA does just enough work not to get caught and collect fees, but does not otherwise actually serve data to users, thus not contributing to security, yet without actively trying to steal funds.

Solution: Make the future DAs that watch the current DAs as indistinguishable from regular users as possible. We can use measures such as:

- Have the watchers be not other members of the current committee, but yet-unrevealed (yet self-identified) members of a future committee.
- Minimally cooperating with other DAs to avoid punishment while denying data to actual users, or otherwise being malicious in modifying their behavior based on who is talking to them—avoided by making DAs indistinguishable from regular users by (1) having unrevealed DAs from a future period watch DAs of the current period, (2) designing the query protocol and the access patterns so it’s hard to distinguish a DA from a user, e.g. by having queries go backwards as well as forward or having fixed granularity, (3)

optionally, but probably importantly if a client makes more than one request, using Onion routing to make requests indistinguishable by address of origin (at the cost of extra latency and reduced throughput).

2.3.7. DA who just relay queries

Issue: a lazy DA may “just” relay queries to other committee members without doing any work of its own, counting on at least 67% of the other DAs to do the job, and collecting fees without doing any real work, without actually contributing to network security, while using up limited committee slots.

Solution: Make it more costly to relay queries than to actually do the work, by having each DA carry a different fragment of the data (using some erasure coding), such that a DA would have to query over one third of the other DAs to reconstitute its fragment on the fly.

2.3.8. DA reneging on what it claims to publish

Issue: DA telling SO his data was accepted but not actually including it.

Solution: The SO requires a commitment from each DA to keep the data available. The SO publishes the commitments in the message that relates the shards together. DAs who don’t provide service get no reward, and further get their stake slashed if they did promise service but failed to deliver.

2.3.9. DA refusing service to SO

Issue: DA for some reason legitimate or il- will not serve SO.

Solution: Allowing SO to get away with only 67% operators having the data—though that will cost *more* (see below).

2.3.10. SO trying to skimp by not contacting all DAs

Issue: Since only 67% of DAs are required for the data to be considered available, a cheap SO might be tempted to only contact 67% of DAs and save on 33% of fees.

Solution: Structure DA payment so SO is just as interested in as each DA in there being as many DAs who know the data as possible.

For that, the network has a payment token (can be the same as the staking/governance token, but could be different), say SKY. Payment happens in some dedicated blockchain and/or in as many state channels. SKY Protocol layer Accounting ensures everyone is paid.

The SO initially overpays about 3 times the expected market rate. The actual fees increase quadratically rather than linearly with the number of DAs who sign past 2/3 of the committee, from something that is already above costs to the expected price. However, out of the initial

margin, the SO also gets a rebate equal to the fee paid. The remnant is burned after a partial rebate is claimed, or no rebate is claimed and some timeout is reached. Burnt fees are unavailable to either DAs or SO—they are sent to a treasury address, to be used to improve the network and/or for charity.

This rebate mechanism ensures that the interests of the DAs and SO are aligned in getting as many DAs in the committee as possible to sign off the data and make it available. For the DAs to get their bonus and the SO to collect their rebate, the SO must provide sufficient signatures, thereby showing that he made the data available to all shards. If he fails to do it on time, members of the committee in turn get the opportunity to collect the data from everyone, compute the missing shards, get a sign off, and collect part of the rebate for themselves.

2.3.9. DAs censoring data

To ensure that the data is censorship-resistant, the data is sent separately to each DA, in uncorrelated blocks. The blocks are only correlated when the decryption key is provided as well as the correlating data.

Payment is made to individual DAs each having their price threshold (and pricing themselves out if the price reduction from the extra signature isn't worth the increased fee), or to the SKY Protocol agreeing on a price (which requires some price consensus mechanism—not covered in v1.0).

DAs can't censor data post hoc: not only would that require >1/3 of the committee making real-time decryption and consensus against posting the correlate revelation; it wouldn't prevent the posters from publishing the correlation directly onto the consensus layer; losing the rebate (but DAs also lose their bonus).

Future Improvement: Payment for a block of data happens in advance at a network-wide level (as opposed to individual DA) for a commitment to the entire set of correlated blocks. The SO provides each DA with a zk proof of payment (plus any extra incentive if required) so the DA knows he'll be paid without knowing about what data exactly or with what bonus. When revealing the correlate, the SO triggers the rebate and bonuses and validations—seconds or minutes later (the longer we wait, the more effectively censorship-resistant, but the more latency is added). To avoid post-reveal censorship, validators have to follow the DAs even before the correlates are revealed, and sufficient time should be allowed for it. Bandwidth has to be reserved for all the validators, plus traffic for the clients, the other DAs acting as guardians, etc.

2.3.10. Censorship by Spam Flood

Those who want to prevent some data from being published on time can flood the DA network with spam, consistently paying higher fees than the honest publisher. This is an expensive attack, but it can work well for a short while against an unprepared victim.

The short term solution is that a prepared victim of such attack should be ready to outbid the censor to get their transaction published within a large enough window. The attacker has to outbid every block, costing him a lot of money, whereas the victim only has to win one auction for space to parry the attack, which is slightly costlier than usual—but thousands of times cheaper than the attack if there are thousands of blocks in their window.

A long term solution is for a potential victim to reserve bandwidth on the DA network—if not for all their traffic, at least for a minimum baseline of important data. A future version of SKY Protocol will support such bandwidth reservation.

2.3.11. Censorship by Data Inspection

Some dishonest DAs might collude together and with bad governments to censor the transactions of SOs based on the content being posted, by decoding it in real-time, and refusing to attest having seen it if the data displeases the censors.

To make such attack harder, the SO will in the future be able to encrypt the data and get it attested before publishing the decoding key directly on the (hopefully censorship-resistant) Layer 1 for his side-chain. Using a "*non*-systematic code" for the erasure encoding (e.g. a variant of Reed-Solomon where the text is the coefficient of the polynome, not its first few values) also makes it harder for censors to inspect the data in real time to reject it.

2.3.12. SO and complicit DAs trying to get unavailable data underwritten

The ultimate attack is for a dishonest SO who controls a lot of DAs (though less than a third) to try to get a block underwritten by the DA network even though it is effectively unavailable, thereby succeeding at a data availability attack, preventing validation (if using interactive proofs) and locking user assets in (even if using non-interactive zk proofs).

The worst a SO and complicit DAs can do while controlling under 1/3 of the network is to get 2/3+ε of the network to underwrite the block by excluding one third of honest DAs, then make their own 1/3-ε among the underwriters never respond to actual validator queries, even though they can reply to all challenges from anyone doing sampling checks.

The solution is that the way data is sharded between DAs must always enable reconstitution of the data from any identified 1/3 of honest nodes actually making their shard available. Note that

dishonest DAs can fail to make any data available, but cannot provide incorrect data, thanks to message authentication via hashes and signatures.

Erasure coding, as used by other Data Availability networks, can be used with the optimal parameters as above.

3. Long Term Vision

3.1. Managing Blockchain Complexity Away

3.1.1. Simple for End-Users not to Computers

Cryptocurrencies will be successful when unknowledgeable end-users can “just” use it and not have to care about all the complexity underneath—though complexity underneath there must be. They should express their intents in terms of transactions they care about, and all the details should be handled and hidden away by their local software coordinator—whether using “AI” or not, running on their own trusted machine.

Suppose you want to purchase some service online. You’ll go to your favorite online mall, specify the kind of service you need, search for adequate providers, choose the one with the best offer for your needs, negotiate the exact terms, sign a contract, send conditional payment in, wait for delivery, check the results, settle the payment. That’s already complex enough (10 steps!, and that’s as much as you want to care about. Often you’d gladly delegate away even those steps. Arguably, Amazon has made a lot of money out of simplifying or grouping 3 or 4 of those steps away in the common case.

Now to do the same purchase on a Decentralized Platform, in addition to the visible steps above, you will be paying in your own favorite token, the service provider will be receiving payment in their own favorite token, the mall in yet another token. On the way, a smart contract will be drafted and signed on a suitable blockchain (that may involve one more token if the previous ones weren’t on smart contract capable blockchains), exchanges may take commissions for token swaps. Many additional services may be invoked and paid for underneath using micropayments: insurance, data access, token intermediation, watchtowers for state channels and side-chains for each party, storage of each participant’s ongoing transaction state in encrypted form on multiple redundant data centers across the world, etc.

There can be a lot of underlying complexity in what for a human should remain a simple transaction, and all of it should be automatically managed by the respective participants’ software agents; absolutely none of them should ever surface to an end-user anymore than they do for services using centralized applications. And yet, it should all happen behind the scenes, because it is all required for the decentralized experience to be as seamless as the centralized

one. Furthermore, it should all be written in ways robust enough to resist numerous government-backed adversaries trying to stop you or steal your assets—because these adversaries exist.

3.1.2. Components not Applications

No one is ever going to write a complete application that does all that Amazon does, and furthermore handles each of tens or hundreds of blockchains, their "wallets", and all the extra details we described above, not to mention those we omitted. Nor should they. It doesn't make sense. It's too big and complex to affordably be written in a way that will be robust enough to resist an attack. More importantly, it's a bad division of labor wherein a central Application developer has incentives unaligned with those of the end-users whose software he controls.

The solution is a modular software platform, in terms of *components* that are finer grained than complete *applications*. A modular software platform includes lifts limits to how much information can be processed by each developer, and increases development efficiency. But more than issues with *information*—a modular software platform solves issues with *incentives*. Corporate managers often disregard matters of incentives, because they live within an authoritarian world where incentive alignment issues are supposed to be solved by discipline enforced by the Human Resource department (even though the corporate hierarchy only solves the most obvious incentive issues with a coarse tool while introducing plenty of other issues of its own). However, decentralized system developers live in no such bubble, under no such illusion. Incentives are everywhere. They are ever present in the "cryptoeconomics" of blockchain protocol (which is really just microeconomics applied to cryptocurrency networks). And they are oh-so-present in rewarding attackers of protocols that are badly designed, implemented or operated—and punishing their users and authors.

In a modular software platform, software is necessarily be made of many components made by many different programmers and companies with interests that potentially conflict with each other's and with the users', unless kept aligned by continuous purposeful action by the participants. Each component must be run inside a software container that ensures it cannot deviate from the declared aligned interests, and appropriately validates all its communications with the system or other modules. The many components must then be assembled in a way that fits the problem at hand. In simple cases, this can be done by hand; in less simple cases by an orchestrator written by hand; in yet less simple cases, by specifying the problem by hand then having a solver find a solution; in more complex cases by having a program identify the problem constraints, then a solver find a solution; etc. In the more complex cases, an "AI" may find the solution—but then there is the issue of ensuring that the proposed solution makes sense—especially in the context of an adversarial environment as must be assumed in any decentralized context.

3.1.3. Automatic Implicit Composition

For instance, one “application” could be an auction for real estate, wherein the “toplevel” component explicitly composes together (a) a general purpose second-price auction component with (b) a real estate NFT component connected to (c) a specific real estate registrar component. A specific *execution* of that application may in turn explicitly use (d) a component moving most transactions to Hydra state channels to make them faster and cheaper, that the original “application” never needed to know about. The seller and bidders behind those state channels may in turn involve (e) payment intermediaries possibly on multiple blockchains, possibly with centralized on/off ramp, (f) currency swaps on a decentralized exchange, with (g) optional insurance against the volatility in the swap should the bid fail, etc. Once again none of the components involved need know much about the others, much less explicitly connect to them.

And all of it should be a smooth experience for users, who should never have to explicitly compose all these components together, or be even aware of the complexity involved—they should just be selling and bidding, each using their own choice of token while paying minimal fees to pay all the many intermediaries. Even the author of the toplevel application component never should have to foresee all the pieces involved during any execution of his application, much less all the infinite possibilities involved during every future execution of his application. maybe long after he dies, involving technologies that didn’t exist while he was alive, yet still using his pristine unchanged well audited and trusted original application. The system will automate the correct composition of all components involved—or rather, it will be automated by the collection of independent systems each running as the personal trusted agent of one participant and only knowing what concern this participant and nothing else. The auction house using Cardano may never know that the seller actually pockets his earning in Bitcoin while the winning bidder was actually using Ethereum, and other bidders were using other blockchains or centralized services. None of the intermediaries need see more than strictly needed to operate their part—they may not even know what they are doing is part of an auction.

3.1.4. Components: Neither Libraries nor Services

A component is different from either a “service” or a “library” as can be found in today’s Operating Systems like Windows, Linux or macOS, Android or iOS:

1. A component comes with metadata on how to properly compose it, suitable for use by a controller using automated reasoning. It is neither pre-assembled like an application or service, nor for use only by experts who will use it while assembling an application or service like a library. It is to be dynamically assembled on the fly by the system orchestrator.

2. A component comes with its own security capability restrictions, as reviewed by security auditors. These capabilities do not leak to the entire application as with a library, or to an entity having access to all data by all users as with a service.
3. Components don't usually involve a system-global server as with services, but one or many well-scoped instances as with libraries. Yet they may involve shared state in a way that libraries cannot achieve without requiring a service.
4. Components can control other components or be controlled by them in ways not practical in traditional architectures (though theoretically possible using nested containers or special system calls usually reserved for debugging or emulation).

A component-based architecture is thus quite different from the centralized operating systems of today designed to support centralized applications. See F.R. Rideau's PhD thesis (undefended) for more details.

3.1.3. The Road to a Future Platform

SKY Protocol is aiming at a world where all Apps are DApps, where all DApps are modular, and where developers can distribute and sell composable App components as well as complete integrations thereof into monolithic Apps—all of it securely.

Now, a modular software platform as required for DApps does not exist yet. But all the bits and pieces of it already exist somewhere, and we "only" have to put them all together:

- Emacs is a successful platform made of many components that notably communicate through "hooks" that each can be implemented by many different other components.
- Rees' W7 security kernel, Mark Miller's E language and its successors Agoric or Spritely, have shown the way for how to write components that can each have their own capabilities that restrict its behavior for security purposes.
- The overall capability model of OLPC's Bifröst, that inspired to a point that of capabilities in iOS then Android, can more directly inspire a platform that would be developed in the interests of the users rather than in that of a megacorporation.
- My (François-René Rideau)'s own PhD thesis was about how a reflective architecture allows finer grained components that span several layers of abstraction and have distinct capabilities at each level.
- Our sister team at FormalFoundry.ai have demonstrated how to combine AI and formal methods to automatically find solutions to constrained problems that are correct by construction.

We at SKY Protocol have an entire vision of why and how DApps should be written in a modular way—and not be monolithic DApps, but dynamic assemblies of components; assemblies driven not by corporations exploiting the end-users, but by software agents that earnestly represent their interests.

Implementing our goal will take a lot of funding. But our vision is not a mere utopia disconnected with reality; quite the contrary, it is a perspective that informs us about reality, and tells us not only of a better place to be that we can see, but also of the road to get there, of the low-hanging fruits along the way, of the pitfalls to avoid and shortcuts to take that others can't see.

We see SKY Protocol Cardano as a stepping stone to a better wider software ecosystem and a low hanging fruit to getting it bootstrapped: it is a component that fits in the existing non-modular platforms as well as in the modular future; it can securely scale DApps in ways that were not obvious without the modular perspective; it can make money this way and bootstrap the development of the larger vision.

3.1.4. A Marketplace for DApp Components

Eventually we will launch a platform that provides the kind of modularity we want (or work with an existing platform to add the missing features). And then we will have a marketplace wherein DApp developers can create audited components that each safely provide a feature, or offer services that are suitably constrained to be secure.

SKY Protocol's Software Component Marketplace can become *the* place where software and software services are exchanged, enterprise software as well as consumer software and everything in between.

3.2. Key Blockchain Components

XXX EDIT THIS SECTION AND BELOW XXX

3.2.1. Essential Blockchain Services

This setup will allow DApps running on any chain to scale independently of the blockchain they use for finality.

For this to happen, the SKY Protocol ecosystem is based on the following parts:

1. A Mutual Knowledge Base
2. Generalized State Channels
3. Side-Chain markets
4. Account View on UTXO models
5. The Glow programming language

3.2.2. Mutual Knowledge Base (MKB)

The Mutual Knowledge Base (MKB) serves as a universal Data Availability Engine capable of supporting an array of Side Chains and State channels across multiple blockchains, with an initial focus on Cardano. Essentially, the MKB functions as a distributed registry processing transaction outcomes among users without prior history knowledge. It then hashes these outcomes and transmits them back to the current blockchain. Functioning as a parallel network of registrars, the MKB constructs Mutual Knowledge—a publicly accessible record of information accessible to all. Compared to Common Knowledge, Mutual Knowledge offers swifter and more cost-effective scalability, presenting a superior solution for network expansion. Furthermore, the MKB's scalability surpasses that of a consensus blockchain, with individual DApps receiving their own rate-limited network shard. Although the scaling solution still necessitates Consensus (Common Knowledge) at specific junctures, it significantly conserves resources and computation time. Moreover, the MKB serves as a pivotal intermediary between Side Chain Markets, enabling the exchange of transaction data while preventing block-withholding attacks, crucial for Plasma-like side-chain designs. The SKY Protocol Scaling Solution leverages the MKB's capabilities to bootstrap its operations, utilizing a simplified variant where both services are provided by the same entities. Powered by its utility token (SKY), the SKY Protocol MKB maintains its Side Chain Market, accelerating payment to registrars. With Proof-of-Stake (PoS) consensus, the MKB ensures network security, with additional protections such as asset-indexing and continuous defragmentation enhancing overall robustness. An example implementation of the SKY Protocol design showcases its ability to facilitate fast transactions via Side Chains while bolstering security through the MKB. While still in development, this implementation underscores the potential of the SKY Protocol to process high-volume transactions efficiently and securely.

3.2.3. Generalized State Channels

State channels are a layer-two blockchain scaling solution. They allow improved throughput and latency for the main blockchain, by allowing uncontested transactions between multiple parties to settle off-chain, and a quick resolution when a transaction is contested because of the body of transactional cryptographic evidence that they accrue, so that in either event, in-progress contracts can continue under the consensus provided by the main chain.

3.2.4. Side-chain Market

Another approach to bolstering the scalability of a blockchain is through side chains. These chains allow a group of participants to execute a series of smaller transactions off the main blockchain, only committing the initial and final states back to the main chain. Our scaling marketplace endeavors to establish a network of state channel and side chain operators, each overseeing a side chain in a predominantly centralized manner, thus facilitating rapid

transaction recording for specific DApps. Unlike traditional centralized applications, however, each side chain is publicly disclosed as a verifiable data structure, enabling operators to audit one another and hold each other accountable for any instances of failure or abuse.

To ensure mutual accountability, operators within a Side Chain Market adopt the Consensus-as-Court paradigm pioneered by TrueBit and Plasma. Assets are managed on the "main chain," which provides consensus (e.g., Cardano), through a "Smart Contract." This consensus is solely utilized for settlement and dispute resolution in case of double spending or contract breaches, with the Smart Contract code effectively serving as the "smart judge." Participants (operators, registrars, and users) can lodge claims against the Smart Contract, which are inexpensive when substantiated but costly when unsubstantiated. Following a challenge period (and potential challenges), the claims are validated (or invalidated), and if validated, users can retrieve corresponding assets from the Smart Contract based on those claims. In the event of a dispute, this process unfolds automatically, resulting in transgressors being penalized and victims compensated.

Users retain the option to exercise their "right to exit," unilaterally repudiating any unsatisfactory operator and switching to another service provider without incurring switching costs, queries, conditions, or concealed fees. Operators failing to meet their contractual obligations automatically forfeit all their users in favor of remaining operators, in addition to losing their bond. Malicious operators are unable to steal or freeze their users' assets but can merely stall them until they lose their users.

A Side Chain Market can be tailored for each specific Distributed Application (DApp). Despite incurring a marginal additional cost per transaction, with the sharing of capital costs, any number of DApps can operate on the same general-purpose Side Chain Market.

The Side-Chain Market introduces the security assumption that at least one operator is honest. Furthermore, the security of a Side Chain Market necessitates a Data Availability Engine to prevent any "block withholding attack" on the Smart Contract controlling the deposited assets. While any existing consensus blockchain could theoretically serve as such an engine (as in the Ethereum "rollup" design), SKY Protocol opts for a dedicated Mutual Knowledge Base for superior scaling at a reduced cost.

This solution holds significant benefits for Cardano, as it enables rapid transaction processing and enhanced scalability for DApps operating on the platform, aligning with Cardano's vision of providing a secure and efficient blockchain ecosystem. Additionally, the accountability mechanisms inherent in the Side Chain Market contribute to the overall security and trustworthiness of the Cardano network, fostering greater confidence among users and developers alike.

3.2.5. Glow Language

Glow is a programming language tailored for decentralized applications (DApps), prioritizing safety, user-friendliness, and portability across various blockchains. For a Cardano Layer 2 protocol focused on data availability, integrating Glow offers several key benefits. Firstly, Glow's emphasis on safety ensures that newly implemented functions undergo thorough audits, minimizing the risk of vulnerabilities in interactions between users. This aligns with the protocol's objective of providing a secure platform for storing and accessing data. Secondly, Glow's user-friendly design simplifies the development process, making it easier for developers to create DApps that meet user expectations. This accessibility can attract a broader audience to the protocol, enhancing its adoption and usability. Lastly, Glow's portability enables DApps written in the language to run on any blockchain, allowing the Cardano Layer 2 protocol to leverage resources from multiple blockchains simultaneously. This flexibility ensures that the protocol can adapt to the evolving blockchain landscape and reach users across different platforms, further enhancing its utility and scalability. Overall, integrating Glow into a Cardano Layer 2 protocol focused on data availability streamlines development, enhances security, and increases accessibility, ultimately contributing to the protocol's success and growth.

3.2.6. Account View On eUTxO Model.

AVOUM (Account View On UTXO Model) is a technology that combines the strengths of the UTXO (Unspent Transaction Output) and Account models in blockchain architecture. For SKY Protocol, a Cardano Layer 2 focused on data availability, AVOUM offers several benefits. Firstly, AVOUM enhances the security of the blockchain by leveraging the efficiency of UTXOs to check past transactions in parallel, enabling faster addition of new full nodes and thereby strengthening the network's security. Additionally, AVOUM introduces the capability for users to queue multiple future transactions concurrently on the same smart contract, facilitating a more dynamic and robust contract ecosystem. This feature is particularly advantageous for SKY Protocol, as it enables the development of richer and more versatile smart contracts, enhancing the platform's functionality and utility for users. Moreover, by combining the UTXO model for past transactions with an Account view for future transactions, AVOUM enables the implementation of public smart contracts without sacrificing decentralization. This aligns with SKY Protocol's goal of providing a secure and decentralized platform for storing and accessing data. Overall, integrating AVOUM into SKY Protocol enhances its scalability, security, and functionality, positioning it as a competitive and innovative solution in the blockchain space.

4. Governance

SKY Protocol's governance model operates through a decentralized autonomous organization (DAO) structure, empowering participants to influence the platform's development and decision-making processes. Participants can vote on various proposals and initiatives through the SKY Protocol (SKY) governance mechanism. These proposals encompass a wide array of topics, including protocol upgrades, fee adjustments, parameter changes, and resource allocation.

However, there are parts of the roadmap that are not dependent on the users' preferences and that can't afford to be built in a bottom-up way. Thus these parts of our roadmap will not be submitted to any kind of vote. Participants can cast their votes on these proposals using SKY tokens, with each token representing a proportional voting power.

The governance process is designed to facilitate community consensus and ensure that the platform's evolution aligns with the collective interests of its stakeholders. The voting process will be open source and on-chain for complete transparency and verifiability of all governance votes.

5. Development Roadmap

Milestone	Timeline
Ecosystem Partners	**Q2 2024**
Cardano L2 Testnet	**Q3 2024**
Audit	**Q3 2024**
Cardano L2 Mainnet	**Q4 2024**
Cardano L2 Expanded Features	**Q1 2025**
SKY Protocol Expansion	**Q2 2025**

5.1 Testnet deliverables

For deploying our L2 testnet, we will use many components from our already built Glow Language. This is publicly available on Github.The next milestones are the following.

Deliverable #1: Contract that interacts with a Centralized Data Availability Operator. It will be a simple Cardano contract that depends on specific data having been published on a *centralized* Data Availability service. For instance, we will write a contract that pays a certain amount if a preimage to an agreed-upon hash was published on the Data Availability service. This milestone will thus demonstrate how a simple Cardano DApp can make use of a Data Availability service.

Deliverable #2: Contract that interacts with a Decentralized Data Availability Committee. The third milestone will be a contract that depends on specific data having been published on a decentralized Data Availability service. We will modify the DApp above to work with a *decentralized* service. The DApp will be largely unmodified, but the data availability validation library it uses will be more sophisticated.

5.2 Mainnet deliverables

Deliverable #3: End-to-End integration of contract using the Data Availability Network.

This can be considered as the SKY Protocol MVP. What we want is the data availability engine and the ability to publish data. This will already allow all DApp developers to have a functional scaling solution. We might only have one topic available.

5.2.1 Stretch goals for SKY token sale

- Multiple topics
- Switching to proof-of-stake instead of proof-of-authority when SKY Protocol has reached a significant amount of users
- More versatile validation rules for specific side-chains/shards and topics: update of acceptance script, choosing between stateful or stateless interactions, configuring the remanence (duration that data remains available)

5.3 Expansion - phase 1

- Portability to other chains with a native bridge to the existing SKY Protocol.

5.4 Expansion - phase 2

- Development of a flourishing Dapp ecosystem using SKY Protocol's scalability with geographic adjustments for the countries that need to work with regulatory constraints.
- Implementation of our bug-bounty and grant system.

6. Tokenomics Introduction

SKY is the utility token for the SKY Protocol. A detailed explanation of SKY tokenomics will be released as a separate document and the link will appear in this section.

7. Team

Our team is comprised of seasoned blockchain and Cardano developers, with a track record of innovation and leadership in the industry. Notably, our team spearheaded the development of the Glow Language, previously commissioned by IOHK, and pioneered the Account View on UTXO, or AVOUM, technology.

Recognized for our contributions, we were honored as Catalyst grant recipients for our work on Formal Verification for Glow and our comprehensive study of AVOUM's potential impact on Cardano. Moreover, our team has shared insights and expertise at various Cardano Conferences. Charles Hoskinson commented on our AVOUM model and his thoughts on the technology in this video.

Beyond the realm of Cardano, our team has made contributions to the academic discourse, publishing papers on formal verification and blockchain technology. Those publications can be found here: https://mukn.com/publications-from-our-team/

Our team has also earned grants at several Catalyst funds, and has developed other pieces of technology for Harmony One, Filecoin, XRP / Ripple, Nervos, Sequentia and has a live crypto-currency payment app on Salesforce.

In addition to its usual team, SKY Protocol has also a partnership with Blink Labs, who are known for their extensive knowledge of Cardano and their proven track record as developers in this ecosystem.

Our superpower is fluency in many domains ranging from theoretical computer science, programming language design and implementation, logic, reflection, distributed systems, network protocols, cryptography, game theory, economic mechanism design, cryptoeconomics, finance, law, adversarial thinking, and more. Our secret weapons are the programming languages Gerbil Scheme and Cubical Agda, enabling us to achieve what others cannot. Our team will grow in numbers after the token sale of SKY Protocol on Cardano, and this document will be updated.

Faré / François-René Rideau - Lead Architect

Faré is the President and Chief Scientist of MuKn, a consultancy creating innovative decentralized solutions for both public and private blockchains.

President and Chief Scientist of MuKn, François-René “Faré” Rideau has more than 25 years of experience building programming languages and distributed systems. He notably proved the correctness of a (centralized) payment protocol early in his career.

Alumnus of École Normale Supérieure, Université de Nice, and Télécom Paris, Faré went to the United States and worked as a Senior Developer for companies such as ITA, Google and Bridgewater Associates.

While working in the industry, he notably maintained and rewrote ASDF, the build system at the heart of the Common Lisp open source community; he also kept publishing academic papers and speaking at programming language conferences.

Eventually, his interests in economics and software security converged with his experience in open source software and formal methods and he started working on Layer 2 solutions for the blockchain. Faré was also one of the original developers of the Alacrity language that later was forked to become Reach on Algorand.

Other technologies designed by Faré but not previously referred to in this whitepaper include:

- **UVOAM:** Enables private and composable UTXO-style transactions on account blockchains.
- **Sigurity:** Provides programmable multisigs indistinguishable from regular signatures to outsiders, yet fully authorized, audited, and logged within the corporation.

- **IWillPersist:** Ensures that DApp clients' data persists securely and privately on redundant blinded remote servers.
- **Durabo:** Empowers users to control their own uncensorable decentralized message feeds.
- **Decenstake:** A censorship-resistant leaderless Proof-of-Stake consensus algorithm.
- **TrieZip:** Accelerates the Merkleization of large data structures by 10x.
- **EVMBatch:** Allows atomic batching of transactions in a single group transaction.
- **MetaCreate2:** Ensures that contracts can have the same address on all EVM blockchains.
- Efficient implementations of Yield Farming, Automated Market Making, Lotteries, or ERC20 contracts.
- **Legicash:** (Designed at a previous startup) Automates the generation of "proof-of-fraud" machinery for optimistic side-chains.

Gauthier Lamothe - Head of Operations

Gauthier specializes in business development and communications. He has been a head of operations and CEO of a few successful companies in France and has worked as a team leader and business developer in previous blockchain projects for MuKn and the Free Republic of Liberland.

Gauthier is an expert in all areas of media, and is passionate about using blockchain to solve global injustices. Former film producer, Gauthier produces MuKn's "Mutual Knowledge" Podcast and educational information on blockchain technology and business, interviewing the world's leading experts on using blockchain technology for social change.

Areas of Gauthier's previous works include decentralized justice systems, tokenization of governance, and use of cryptocurrency for decentralized systems of freedom.

Doubling his career with psychotherapy and coaching, Gauthier acts as a trainer and advisor for many companies (in France or in other countries) in fields such as biotech, insurance, wellbeing, banking, nuclear physics, fast-food, and goods distribution.

Alex Hochberger - Tech Advisor

Alex M. "The Hoch" Hochberger has over 20 years of experience integrating technology and business systems to drive real growth. He notably spent an extensive amount of time building and advising startups from formation to exit. His first startup was formed with his MIT roommate (sold 7 years later) through doing technology turnarounds of portfolio companies at Z9 Ventures, a Miami-based venture group.

One of the original MiamiTech “OGs,” Hoch was a mentor with Incubate Miami, the original Miami Accelerator. Now he is an advisor with Cryptan Labs Web3 and Blockchain Accelerator in Miami.

Alex’s career has included building out and configuring Enterprise and Startup versions of CRM, ERP, and Data-storage systems, from the Web1 Dot-com Bust through the Web3 era, with expertise using technology to solve real world problems. His stints have included CTO and CMO roles in consumer finance, health care, and consumer product space.

Last but not least, he is the CEO of Web3 Enabler, offering crypto-payment solutions on Salesforce

Zoe Braiterman - Data Safety Expert

Zoe Braiterman has experience ranging from cyber security, data science and system architecture to product development.

She is strong with an extensive experience as a security consultant and research associate for PurePoint International, Analyst and Lead Data Architect for multiple companies and an active prominent member of the OWASP Foundation.

Passionate about helping startups to scale, she also has experience as a teacher and business manager. She now secures the whole MuKn information ecosystem.

Donald Fisk - Developer

Donald has a BSc in Physics and Astronomy from Glasgow University,and an MSc in Telecoms Engineering from Kings College, London.

Donald began programming in 1981, and became interested in Artificial Intelligence and Lisp shortly afterwards. Lisp is his favorite language, but he has also programmed in C, Java, Python, Prolog, Pascal, various assembly languages, and more recently in Solidity.

He has developed AI rule-based systems for fault recognition and routing in telecommunications systems, musical score generation from MIDI input, phonetics, and web page layout, and also used Lisp for document processing, and to develop a visual dataflow programming language, Full Metal Jacket.

He has had papers published on MORSE, a collaborative filtering system for movie recommendation, and on Full Metal Jacket. He has worked mostly in research and development, and is inventor/co-inventor on six patents in telecommunications, computer graphics, and musical score generation.

He has a personal web page at http://fmjlang.co.uk/, which has links to his papers and patents.

His blockchain experience started in 2021 when he worked on a few development projects for MuKn, including a data availability engine.

Marcin Grzybowsky - Developer

Marcin has been working professionally in software development for more than 15 years.

After using Agda and Idris for personal projects, he became captivated by dependently typed languages and Homotopy Type Theory and has been using these technologies for the benefit of commercial projects Since 2018.

At MuKn Marcin is notably tasked with the implementation of the Formal Verification functionality of Glow, and the development of machine-checked proof of its properties, but his work also includes the development of other technologies in our own R&D lab.

Multiple Laureate of the Polish Physics Olympiad (2004,2005), Marcin has experience in many different fields ranging from Artificial Intelligence, Domain Specific Languages, and has also cofounded startups.

Marcin is also experienced in the field of AI, involved in the FormalFoundry project, that uses formal methods to prove the safety of AI applications and AI to improve the usability of formal methods, offering technologies that combine machine learning and software verification to the wider public.

Alexander Smart - Chief Legal Officer

CEO of MuKn, Alexander Smart has a B.A. from UChicago with an emphasis in Political Science and a minor in Economics, and a J.D. from Pepperdine Caruso School of Law. He is licensed to practice law in California, New York, and Massachusetts.

Alexander has always thought fast, but learned to think deep and sharp at UChicago. After studying law at Pepperdine, he spent nearly fifteen years guiding executives and decision makers through litigation, in matters ranging from shoplifting and speeding tickets to multi-forum international investment bank disputes.

His practice honed his ability to quickly assimilate and master new information, and deliver that information clearly at any level of sophistication. Tiring of courthouses, he found his skills were readily applicable and desperately needed in the blockchain space.

Informed by two years of volunteer work he did in Northeastern Brazil in his college years, he joined MuKn to help use Web3 to give people in every part of the world access to modern financial services.

Peter Hubshman - Chief Financial Officer

Peter is a finance and operations expert focusing on early round startups. With origins in private equity, fund management and leveraged buyouts, in the early 2000's he operated Internet Real Estate Group, a Web 2.0 studio in Boston which successfully developed businesses including Creditcards.com; Phone.com; Luggage.com; Jeans.com and a dozen other early primary domain businesses. There, his pioneering team of engineers created some of the earliest successful affiliate marketing and advertising platforms on the Internet, and were early experts in search engine optimization.

In 2009 Peter became CFO of Digiport Data Centers in Miami FL until its sale in 2013. There he developed business plans and economic models for all of Digiport's spinout startups, including:The collaborative consumption platform, Boatsetter.com (Air BnB for yachts, captains included); SaaS startup, Itopia.com (early cloud-ware for small-midsized professional offices and now, .edu); and artworld upstart, Blackdove.com (a digital arts platform for digital artists, restaurants, residential, and corporate). In 2018 Peter was Management Consultant to, and Interim CFO at TheDroneRacingLeague.com (e-sports, media, and drone technology). There he helped prepare the management team and crew for a successful C round led by a prestigious investment bank in 2019. Later in 2019, he became Interim CFO of Gemic.com (Brand Strategy]]]]]for Fortune 100's). There he led the financial team supporting their successful A round with Bocap, a private equity firm in Finland.

Peter joined MuKn in Q1 2022 as CFO, Co- Founder and a Board Member. He remains a Member of the Board, and is currently Finance Advisor while on assignment as CXO to a new MuKn studio spinout, FormalFoundry.ai (Dedicated to ensuring the correctness of AI models at

scale). Peter studied economics at Tufts University and has a Masters in Public and Private Management from the Yale University School of Management.

8. Security

SKY Protocol embodies an open-source approach, elevating security standards in line with the principles of cryptocurrency. This commitment to transparency and scrutiny mirrors the ethos of decentralization and trust inherent in cryptocurrency networks. With openly accessible code, a diverse community of developers, auditors, and users can meticulously examine it for vulnerabilities, errors, or malicious elements, fostering a culture of collective responsibility and collaboration. Conversely, closed-source models restrict code visibility, limiting the breadth of perspectives available for security evaluation and potentially compromising the integrity of the protocol. By embracing open-source principles, SKY Protocol promotes a secure and resilient ecosystem, aligning with the core tenets of cryptocurrency networks.

Regarding audits, as an integral part of the development process, the protocol engages external auditors to ensure robust security measures are upheld. These auditors conduct independent assessments, identifying potential flaws and conducting thorough code reviews to safeguard the protocol's security and reliability. This commitment to rigorous auditing reflects the protocol's dedication to maintaining the highest standards of security and transparency within the cryptocurrency space.

A. Appendix: Bibliography

A.1. Previous Relevant Publications by Our Team

François-René Rideau, "Chenilles Whitepaper", whitepaper, 2023.

François-René Rideau, "AVOUM: Account-View-on-UTXO-Model", whitepaper, 2022.

François-René Rideau, "Durabo: Unstoppable Message Feeds, 2021", whitepaper, 2021.

François-René Rideau, "Simple Formally Verified DApps—and not just Smart Contracts", EthCC[3], 2020.

François-René Rideau, "Glow Whitepaper", 2020.

Jay McCarthy and François-René Rideau, "Alacrity: A DSL for Simple, Formally-Verified DApps", DevCon5, 2019.

François-René Rideau, Language Abstraction for [V]erifiable Blockchain Distributed Applications, IOHK Summit, 2019.

François-René Rideau, "Composing Contracts without Special Provisions — using Blockchain History" Hackernoon, 2019.

François-René Rideau, What do Formal Methods actually Guarantee?, 2018

François-René Rideau, "Binding Blockchains Together With Accountability Through Computability Logic", LambdaConf 2018.

François-René Rideau, "Why Developing on Blockchain is Hard? - Part 2: Computing Proper Collateral", Hackernoon, 2018.

François-René Rideau, "Why Developing on Blockchain is Hard? - Part 1: Posting Transactions", Hackernoon, 2018.

François-René Rideau, "Legicash: Binding Blockchains Together through Smart Law", Legicash Whitepaper, 2018.

François-René Rideau, "Climbing Up the Semantic Tower — at Runtime", Off the Beaten Track Workshop at POPL, 2018.

A.2. Other Relevant Publications

Matthew Frehlich and Vishesh, "A First Principles guide to Data Availability: Part 1, Demystifying the Data Availability hype in web3", 2024

"Flashbots Transparency Report", February 2021.

Dan Robinson and Georgios Konstantopoulos, "Ethereum is a Dark Forest", 2020.

"Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges", 2019.

Mustafa Al-Bassam, "LazyLedger: A Distributed Data Availability Ledger With Client-Side Smart Contracts" (Celestia Whitepaper), 2019.

Joachim Zahnentferner, "Chimeric Ledgers: Translating and Unifying UTXO-based and Account-based Cryptocurrencies", March 2018.

"How the winner got Fomo3D prize — A Detailed Explanation", SECBIT Labs, August 2018.

Joseph Poon and Vitalik Buterin, "Plasma: Scalable Autonomous Smart Contracts", 2017

Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2009.

A.3. To be added

https://x.com/inputoutputhk/status/1366838241580703744

https://iohk.io/en/blog/posts/2021/10/29/mithril-a-stronger-and-lighter-blockchain-for-better-efficiency/

https://cexplorer.io/article/what-is-the-future-of-cardano-scalability

https://github.com/ethereum/research/wiki/A-note-on-data-availability-and-erasure-coding

This page is intentionally left blank.