Part 1

* 1. Sharding through random sampling or “quadrating sharding” is an interesting way to solve the scalability problem. You create committees from a shuffled list, and then choose 100 validators to be the committee that will verify a block – each validator in a committee publishes a signature when they verify the block, and the rest of the network need only verify the signatures themselves, not the blocks. However, this comes with trade-offs compared to traditional bitcoin or Ethereum. If some malicious person or group can take control of or turn off any set of nodes they desire at any given moment, they can break committees. This means too, that only some of the malicious nodes can be traced to the attack, therefore the overall penalty (losing their stake) is reduced in this manner as well. There are also issues with data availability if we try to increase the amounts of data, so we limit this.

Another way to solve this would be to compromise decentralization for a centralized high-TPS chain. In this method what we receive in terms of speed and scalability, we give up in decentralization, we do not get to see every transaction and therefore we do not see what got censored by the block producers. It is also much harder to verify why a block got rejected, we wouldn’t be able to tell if it was invalid OR if they were published too slow.

* 1. Starknet is based on the turing-complete, Cairo language. Instead of using circuits, a user can define any business logic, have it proved off chain, and then verified on chain. With features like Generic Proof of Service, they allow for any user to send their code in, to prove, then verify it on chain. A plus of this service is that a single proof can be generated for groups of applications (that have nothing to do w. each other), reducing the gas costs of proving. Starknet has a very robust team behind them, with a clearly defined roadmap and an ever-growing community of folks wanting to learn how and build on top of their platform.

zkSync is another rollup solution that promises low fees, and otherwise a very similar experience to mainnet Ethereum. Some differences right off the bat are that StarkWare is based on STARKS and ZKSync off SNARKS. Starks can be generated about ten times faster than snarks. While STARKS are trustless, the ZKSync will require users to rely on a trusted setup ceremony that was performed around the time that ZKsync was founded in 2019. Zksync has a lead on EVM compatibility compared to the Starkware team, with use of the zkSync compiler, whereas the Starkware team is currently working on a Cairo to solidity translator. Solving data availability also sets these teams apart, zkSync relies on zkPorter – a sharding system in which there is a “master shard” to always ensure data availability, and then any other shard onward can choose different schemes concerning data availability. Starkware’s answer is Volition – which allows the choice of rollup scheme and offchain data availability with each txn. The offchain availability is provided by a centralized committee.

It is difficult to say which will win, but it is pretty clear that starkware has a year headstart, a very public team of the smartest cryptographic minds in academia, and a whole lot of funding (valuation is sitting at ~2 bln right now). I think the winner of the race depends on whether or not Starkware can figure out their cairo translator – the ability to be EVM based cannot be understated given the growing pace of this space. Adding the hurdle of learning a new programming language like Cairo may stifle the onboarding of new developers, since ive tried to do both, I can tell you that it took me about 2-3 days to setup my cairo environment versus zkSync (took less than an hour, but longer than the 2 min they advertise on their website.

Part 2

2.1 A bridge application for a token transfer requires a few things: a smart contract on either side of the bridge, a front end to call all the proper methods and contracts, and a backend to display events/listen to activity on either contract. Technically we also need rpc endpoints and things like that but for now assume these are solved. A contract on the starting point of the bridge will serve as a reserve of said tokens OR will create a supply of tokens. A contract on the end side or other side of a bridge will receive a token and either mint if it sees that tokens are being locked on the starting side, or burn if it receives tokens from the other side. This otherside contract has methods that can only be called from the bridge. Next we will need a front end that is packed with all the methods so that users can call each method properly and go back and forth with funds that go from user 🡪bridge 🡪back to user in either direction. Finally we need a backend that listens to these events and broadcasts them to the user, users should be able to see where their funds are moving in real-time.

2.2 AZTEC first and foremost exists to allow for privacy on blockchains – the current state of blockchains like BTC or Ethereum make them not possible for use by large scale institutions, since all information is available to everyone else. AZTEC uses UTXO’s that are similar to BTC. A note is a data structure. A note is composed of a few things: nonce, asset id, value, secret, owner x, and owner y. These details are then hashed using Pedersen scheme, and the product is called a note commitment. An account note is composed of: an alias id, an account public key, and an account spending key. AZTEC then stores all note commitments in a merkle tree called the note tree. All spent notes are included in the nullfier tree. All old note tree roots are included in the RootRoot tree. There are then circuits that allow for different things to happen with these merkle trees and their associated info. There are 5 in total: JoinSplit, Account, Rollup, Escape-Hatch, Root Rollup. Join split allows for notes to be spent, taking in 2 inputs producing two outputs, and updating the note and nullifier trees. The Account circuit lets us transfer the control of notes, only adding to the nullifier tree sometimes. The rollup circuit groups proofs from a set of inner circuits. The escape hatch circuit allows for withdrawals from the network without the use of a relayer (to rollup the tx). The root rollup circuit does what its name suggests, rolls up other rollup proofs.

Besides these circuits, AZTEC uses an array of proofs for proving different types of information about proofs for example greater than or equal to (range proof), a proof that shows that you own something and are trying to swap it for something else (swap proof) like in a trade using zk currency, for general purpose payments like rewards bills/etc they use send proofs to verify who is to receive what and how much, originating assets such as loans use the mint/burn proof, and finally they use proofs to verify a user’s stake in an income stream without revealing their position in order to distribute dividends (dividend proof).

Part 3

Project Name: ZKSTART

Application type: Grant

Proposal Overview:

Scaffold-eth is a very easy to use general purpose starter kit that helps people better understand dapps on the EVM and gives users a stack and framework to quickly learn and deploy apps. ZKSTART would be built using scaffold-eth, it will include standard ZK and CIRCOM libraries, and have basic circuits and proof generation capabilities. This way, the starter kit will be able to easily be integrated in current scaffold-eth builds (one that I mainly work on which is a gnosis competitor, known as Multisig as a service – multisig.lol) as well as serve as a starting point for people to learn about and employ ZK technology. Once piece of feedback received, was to integrate various existing technologies such as semaphore for logins, and interep to manage reputation – these seem like fairly low hanging fruit and if this would be considered grant worthy I would pursue it, otherwise I am still searching.

Scaffold-Eth currently has about 403 people building and earning a work income stream (as per BuidlGuidl). It is run by Austin Griffith who is part of the Ethereum Foundation. Austin regularly attends conferences/hackathons to spread the word about scaffold and encourage users to speed run Ethereum, or rather complete 4 different challenges using scaffold-eth, ranging from minting an NFT to creating and deploying your own liquidity pool. I think this starter kit would have a very direct approach to gaining users, by virtue of curiosity about ZK tech as well as the bump it would get from being a part of the scaffold-eth ecosystem.

The biggest comparisons would probably be Starkware or zkSync, since these are both platforms that foster creation with developer tools versus an actual application. In this regard, a disadvantage of ZKSTART would be that it is extremely behind since I am a single person, and I am not a grad student/professor of cryptography. That said, this project would have the eye of the scaffold-eth ecosystem, and if it’s decent enough, and broad in application it will quickly be adopted by the builders of scaffold-eth, putting a zk-layer ontop of the DAPP manufacturing plant that scaffold-eth is already. A team would quickly form and Austin would help out on the managing front. One advantage over STARKWARE is that this would be fully open sourced dev tooling, an advantage over zkSync would be a stack that I am intimately familiar with as well as a community of builders which I am apart of