Before diving into the technical details and the goals of this work, we will cover a number of topics that are important to understand in the context of this work.

0.1 Decentralized Exchanges

The first thing to understand is the recent rise of decentralized exchanges. Instead of relying on server infrastructure, as traditional crypto asset exchanges do, decentralized exchanges execute the trade logic in a decentralized and trustless manner. The most popular decentralized exchange is Uniswap [**?**], which runs as a collection of smart-contract on the Ethereum [**?**]

0.2 Types of layer-2 systems

0.2.1 Plasma

0.2.2 Optimistic rollup

0.2.3 Validium

0.2.4 zk-rollup

0.3 Merkle Trees

0.4 MiMC hash function

0.5 zkSNARK

0.5.1 brief explaination of how it works

0.5.2 what can be achived

0.5.3 Why its interesting in the blockchain context

0.5.4 Why zkSNARK was chosen for this usecase

0.6 Explaining Gas

0.6.1 Difference gas amount / gas price

0.7 Replay Attacks

A common security consideration to make in cryptographic protocols are replay attacks. A replay attack occures when a valid cryptografic proof is maliciously resubmitted, and no security checks are in place to invalidate the resubmission. When making a transaction in a blockchain network, the transaction is signed with the users private key. Miners that receive transactions check the signatures validity, which authorizes the transaction as the user has access to the addresses private key. However, a malicious actor could store these signed transactions and resubmit them to the network. Since the signature is still valid, a mechanism is needed to invalidate transaction signatures, once they have been included in a block and thereby executed. In

most blockchain systems, like Bitcoin or Ethereum, this is solved by tracking a nonce for each address. Miners extract an addresses current nonce by indexing the entire blockchain, simply counting the number of confirmed transactions. Among other things, the nonce is part of the signed transaction. When a miner checks the signature, it is also checked if the nonce set in the signed transaction has been incremented by one, when comparing with the nonce that was extracted. This invalidates a resubmission of the signed transaction and prevents replay attacks.

0.7.1 Transaction Replay Attacks in zkSwap

The way transactions are executed in this system is quite different, compared to the blockchain systems described above. As we will explorer, most verifications and computations happen off-chain and are then verified on-chain with a single proof, that

When making a transaction, which can be a trade, a deposit or a withdraw, a user signs the transaction details and sends them to our off-chain entity, the aggregator. The signature is then checked in a zkSNARK circuit, along with the nonce. The resulting proof can then be used to verify the correct execution of the zkSNARK circuit, as an effect verifying the correct. While the signature check happens off-chain, the mechanism of preventing replay attacks is comparable to most blockchain systems. By checking if the nonce signed in the transaction details has been incremented, we ensure the signed transaction is invalidated when resubmitted.

0.7.2 Proof Verification Replay Attacks

Another consideration to make, is the submission and verification of the zkSNARK proof. It also could be resubmitted, thereby breaking the protocol if no measures are in place to prevent this. When verifying a zkSNARK proof, we're essentially proving the execution of the circuit was done correctly, and that the resulting outputs where computed by running the circuit. In this system, the zkSNARK circuit is used to ensure correct state transitions tbc