A Gas-Efficient Superlight Bitcoin Client in Solidity

Anonymous Author(s)

ABSTRACT

Superlight clients enable the verification of proof-of-work-based blockchains by checking only a small representative number of block headers instead of all the block headers as done in SPV. Such clients can be embedded within other blockchains by implementing them as smart contracts, allowing for cross-chain verification. One such interesting instance is the consumption of Bitcoin data within Ethereum by implementing a Bitcoin superlight client in Solidity. While such theoretical constructions have demonstrated security and efficiency, no practical implementation exists. In this work, we put forth the first practical Solidity implementation of a superlight client which implements the NIPoPoW superblocks protocol. Contrary to previous work, our Solidity smart contract achieves sufficient gas-efficiency to allow a proof and counter-proof to fit within the gas limit of a block, making it practical. We provide extensive experimental measurements for gas. The optimizations that enable gas-efficiency heavily leverage a novel technique which we term hash-and-resubmit, which almost completely eliminates persistent storage requirements, the most expensive operation of smart contracts in terms of gas. Instead, the contract asks contesters to resubmit data and checks their veracity by hashing it. We show that such techniques can be used to bring down gas costs significantly and may have applications to other contracts. We also identify and rectify multiple implementation security issues of previous work such as premining vulnerabilities. Lastly, our implementation allows us to calculate concrete cryptoeconomic parameters for the superblocks NIPoPoWs protocol and in particular to make recommendations about the monetary value of the collateral parameters. We provide such parameter recommendations over a variety of liveness and adversarial bound settings.

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper;

KEYWORDS

template; formatting; pickling

1 INTRODUCTION

Interoperability is the ability of distinct blockchains to communicate. These *cross-chain* interactions can enable useful features across blockchains such as the transfer of asset from one chain to another (one-way peg) and back (one-way peg). To date, there is no commonly accepted decentralized protocol that enables cross-chain transactions. Currently, crosschain operations are only available to the users via third-party applications, such as multi-currency wallets. However, this treatment is opposing to the nature of the decentralized currencies.

In general, crosschain-enabled blockchains A, B would support the following operations:

- Crosschain trading: Alice with deposits in blockchain A, can make a payment to Bob at blockchain B.
- Crosschain fund transfer: Alice can transfers her funds from blockchain A to blockchain B. After this operation, the funds no longer exist in blockchain A. Alice can later decide to transfer any portion of the original amount to the blockchain of origin.

To perform crosschain operations, there must be some mechanism to discover from chain A that an event occurred to chain B, such that a transaction occurred. A trivial way is to participate as a node in both chains. This approach, however, is very impractical because a sizeable amount of storage is needed to host for the entire chain, as they grow with time. At the moment, Bitcoin chain spans roughly 245 GB and Ethereum has exceeded the 1 TB. Naturally, not all users have the convenience to accommodate this size of data.

An early solution to compress the extensive space of blockchain was given by Nakamoto with the Simplified Payment Verification (SPV) protocol. In SPV, only the headers of blocks are stored, saving a considerable amount of storage. However, even with this protocol, Bitcoin headers sum up to 50 GB of data (80 bytes per header). A mobile client needs several minutes, even hours to fetch all information needed in order to function as an SPV client. Moreover, not all blockchains have small-sized headers like Bitcoin. Block headers of Ethereum, for instance, are 508 Bytes and in other altcoins the size of each header is several MB. Even with the use of SPV protocol, the process of downloading and validating all block headers can lead to unpleasant user experience.

In order to provide a more practical solution than SPV clients, a new generation of *superlight clients* emerged. In these protocols, cryptographic proofs are generated, which prove the occurrence of events inside a blockchain. Better performance is achieved due to the considerably smaller size of the proofs compared to the SPV protocol. By utilizing superlight client protocols, a compressed proof for an event in chain A can be constructed, and, if chain B supports smart contracts, the proof can be then verified automatically and transparently *on-chain*. Note that, this communication is realized without the intervention of third-party applications. An interesting application of such a protocol is the communication of Bitcoin events to Ethereum.

Related Work. Non-Interactive Proofs of Proof of Work (NIPoPoWs)(ref) is the fundamental building block of our

solution. This cryptographic primitive is *provably secure* and provides *succinct proofs* regarding the existence of an arbitrary event in a chain. Contrary to the linear growth rate of the underlying blockchain, NIPoPoWs span logarithmic size of blocks

Christoglou (ref), has provided a Solidity smart contract which is the first ever implementation of crosschain events verification based on NIPoPoWs, where proofs are submitted and checked for their validity. This solution, however, is impossible to apply to the real blockchain due to extensive usage of resources and important security vulnerabilities.

A different methodology to solve interoperability is a protocol introduced by Summa team. In this work, users waits for a transaction to be buried under several blocks which implies that the transaction must belong to the real chain, and thus be valid. This treatment enables fast and cheap crosschain capabilities. But this solution has not been proved to be cryptographically secure. In fact, an attack has been laid out that makes the protocol vulnerable to non-rational adversaries.

Our contributions. We put forth the following contributions:

(1) We developed the first ever decentralized client that securely verifies crosschain events and is applicable to the real blockchain. Our client establishes a safe, cheap and trust-less solution to the interoperability problem. Our client is implemented in Solidity and verifies Bitcoin events on the Ethereum blockchain.
(2) We prove via application that NIPoPoWs can be utilized in the real blockchain, making the cryptographic primitive the first ever applied construction of secure, succinct proofs in a real setting.
(3) We present a novel pattern which we term *hash-and-resubmit*. This pattern improves the performance of smart contracts in terms of gas consumption by utilizing *calldata* space to eliminate high-cost storage operations.
(4) Optimistic vs non-optimistic constructions ??

Our implementation meets the following requirements:

(1) Security: The client is invulnerable against all adversarial attacks.
(2) Is trust-less: The client is not dependent on third-party applications and operates in a transparent, decentralized manner.
(3) Applicability: The client can be utilized in the real blockchain and comply with all environmental constraints, i.e. block gas limit and calldata size of Ethereum blockchain.
(4) Is cheap: The application is cheaper to use than the current state of the art technologies.

We selected Bitcoin as source blockchain as it the most used cryptocurrency and enabling crosschain transactions in Bitcoin is beneficial to the vast majority of blockchain community. We selected Ethereum as the target blockchain because it is also very popular and it supports smart contracts, which is a requirement in order to perform on-chain verification.

Structure. In Section 2 describe all blockchain technologies which are relevant to our work. In Section 3 we put forth .... In Section 4 we show ... and, finally, in Section 5, we discuss ...

REFERENCES