A gas-efficient superlight Bitcoin client in Solidity

April 28, 2020

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. Our implementation is open source and we demonstrate it is production-ready by providing full test coverage. 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.

Contents

Chapter 1

Introduction

1.1 Motivation

Bitcoin is a form of decentralized money. Before Bitcoin was invented, the only way to use money digitally was through an intermediary like a bank. However, Bitcoin changed this by creating a decentralized form of currency that individuals can trade directly without the need for an intermediary. Each Bitcoin transaction is validated and confirmed by the entire Bitcoin network. There is no single point of failure so the system is virtually impossible to shut down, manipulate or control.

The person (or group of people, as many think) behind Bitcoin, is known by the name Shatoshi Nakamoto. Shatoshi put forth a construction that nowadays some consider one of the most important achievements of our age, all fitting into a 9-page paper. Bitcoin was published in November 2008, shortly followed by the initiation of the Bitcoin network in January 2009 and is the first ever secure and trust-less currency.

One of the by-products of the Bitcoin is blockchain. Blockchain technology was created by fusing already existing technologies like cryptography, proof of work and decentralized network architecture together in order to create a system that can reach decisions without a central authority. There was no "blockchain technology" before Bitcoin was invented, but once Bitcoin became a reality, people started noticing how and why it works and named this construction blockchain. Blockchain constitutes the very core of Bitcoin. Later, it was realized that a currency like Bitcoin is just one of the utilizations of the blockchain technology.

Ethereum was first proposed in late 2013 and then brought to life in 2014. Ethereum is a blockchain network that, apart from its digital currency, Ether, hosts decentralized programs. These decentralized apps (Dapps), or smart contracts, are written in Solidity, the programming language of Ethereum and yield to no single person control, not even to their author. The Ethereum platform is fully decentralized and consists of thousands of independent computers running it. Once a program is deployed to the Ethereum network it will be executed as written, hence the famous phrase: "code is law". Ethereum is a network of computers that together combine into one powerful, decentralized supercomputer. Ethereum is often characterized as the second era of blockchain networks.

With the passing of time, new cryptocurrencies, altcoins as they are called in the cryptocurrency folklore, are created every day. Some altcoins bring new features to the cryprocurrency market and are accepted by the community, even becoming popular. After Bitcoin and Ethereum, the most important blockchains in terms of capitalization are Ripple, Tether, Bitcoin Cash and Litecoin. As of April 2020, there were over 5.392 cryptocurrencies with a total market capitalisation of $201 billion.

A newcomer to this world of distributed coins would possibly expect that there must be some kind of established protocol for all these distinct blockchain to interact; a way for Alice, who keeps her funds in Bitcoins, to transfer an amount to Bob, who keeps his funds in Ether and vice-versa[1]. In reality, the problem of blockchain interoperability had not been researched until recently, and, to date, there is still no commonly accepted decentralized protocol that enables interactions across blockchains, the so-called crosschain operations.

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

- 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.

Currently, crosschain operations are only available to the users via third-party applications, such as multi-currency wallets. It is obvious that this centralized treatment opposes the nature of the blockchain and the introspective of decentralized currencies. This contradiction motivated us to create a solution that enables cheap and trust-less crosschain operations.

1.2 Rationale

To perform crosschain operations between two chains, there must be some mechanism to communicate to chain A that an event occurred to chain B,

[1]The transfer of an amount from one chain to another is called one-way peg, and the transfer of funds back to the original chain is called two-way peg.

such that a payment tool place. One trivial way for Alice to determine if an event took place in chain A and register it to chain B is to participate in both chains. But this way is very inefficient because the storage needed to store the blockchain is sizable, and it grows with time. At the moment, Bitcoin is 242.39 GB and Ethereum has exceeded 1 TB. Naturally, it is impractical for every user to accommodate this size of data.

An early solution to the extensive space of blockchain was given by Satoshi, and is called Simplified Payment Verification (SPV) protocol. In SPV, only the headers of blocks are stored, saving of a lot of storage. However, even with this protocol, Bitcoin headers have the total size of 50 GB (80 bytes each). A mobile client needs several minutes, even hours to download all information needed 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 for different altcoins they span several MB of data. SPV clients lead to unpleasant user experience at some cases, while they are completely impractical in others.

Another idea to make chain A interact with chain B, is to provide a cryptographic proof to chain B that an event occurred in chain A. Secure cryptographic proofs are mathematical constructions that are easy to verify and impossible for an adversary to forge and are broadly used in cryptography and blockchain in particular. In order to be more efficient than SPV, the size of these proof needs to be small related to the size of the blockchain. This way, we are be able to create proofs for events in chain A and send it to chain B for validation. If chain B supports smart contracts, like Ethereum, the proof can be verified automatically and transparently *on-chain*. Notice that no third-party is involved through the entire process.

1.3 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 provide succinct proofs regarding the occurrence of an event in a chain. The size of the proofs is logarithmic to the size of the underlying blockchain, which means that they are growing slowly compared to the blockchain growth rate.

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 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 aspect 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 cryptographically secure. In fact, an attack has been laid out that makes the protocol vulnerable to non-rational adversaries.

1.4 Our contributions

We put forth the following contributions:

(a) 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 to the Ethereum blockchain.

(b) We prove via application that NIPoPoWs can be utilized in the real blockchain, making the cryptographic primitive the first ever applied construction of succinct proofs to a real setting.

Our implementation meets the following requirements:

(a) Security: The client is invulnerable against all adversarial attacks.

(b) Is trust-less: The client is not dependent on third-party applications and operates in a fully transparent, decentralized manner.

(c) 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.

(d) 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 to the community and it supports smart contracts which is a requirement in order to perform on-chain verification.

Some applications that demonstrate the usage of our client are:

- Application #1
- Application #2
- Application #3

1.5 Structure

In Section 2 we description of all relevant background technologies and previous work. In Section 3 we put forth the implementation of our client. In Section 4 we show our cryptoeconomic analysis and, finally, in Section 5, we discuss applications of our client and future work.