Blind Off-Chain Lightweight Transactions (BOLT)

Zcash Foundation Grant Proposal

J. Ayo Akinyele, ayo@yeletech.org

Motivation and Overview

BOLT is a system for conducting privacy-preserving off-chain payments between pairs of individual parties. BOLT is designed to provide a "Layer 2" payment protocol for privacy-preserving cryptocurrencies such as Zcash, by allowing individuals to establish and use payment channels for rapid/instantaneous payments that do not require an on-chain transaction.

BOLT currently exists as a specification and proof of concept implementation partially in Charm by Ian Miers and Matthew Green (the original authors of the BOLT protocol).

Based on my recent efforts in implementing the BOLT protocol in Rust (**work in progress**), the goal of this project is to build on the BOLT library (or **libbolt**) and develop a full node (boltd) that interfaces with Zcash (e.g., a BOLT-compatible cryptocurrency). Both the libbolt library and boltd software are intended to be released as open source to support use cases such as privacy-preserving micropayments.

Therefore, this project will benefit the Zcash ecosystem and further the broader goals of addressing the scalability problems of cryptocurrencies like Zcash and beyond.

Technical Approach

My previous work focuses on the development of the Bolt library (or libbolt) and the core cryptographic components of the BOLT payment channel protocol (e.g., bi-directional payment scheme).

Building on the libbolt implementation, I will provide routines for constructing and parsing the messages required for interactive off-chain transactions with (one or more) remote BOLT participant(s), and the full node will provide necessary routines that can interface with the cryptocurrency node (e.g., Zcash) via its interface.

Boltd will include supporting functionality for channel negotiation, channel funding, activation, payment and closure. This may also include a separate component to monitor the blockchain to manage dispute resolutions between participants.

The boltd node will be connected to the currency P2P network, and will support commands via an RPC interface. I will leverage existing code developed in Go by the Lightning Network project (e.g., lightningd) which uses a gRPC interface to enable clients to communicate directly with the daemon.

Using the bidirectional payment construction of libbolt, there will be five features/functions that will exposed by the bolt deamon to enable payment channels:

1. **Channel Negotiation**: Provides functionality for the Customer and Merchant to agree on the initial balances of the channel, which is denoted by A (customer initial balance) and B (merchant initial balance) respectively. The Merchant provides a public key and signed channel opening transaction to the Customer.

2. **Channel Funding**: Customer transmits the channel opening transaction to the Zcash network, which causes the Customer and Merchant to fund the channel with (A, B) units of currency respectively. This funding is conducted on-chain, and will be conducted using Zcash so as to protect the customer's identity.

3. **Channel Activation**: Once the channel has been funded and the transaction confirmed on the Zcash blockchain, Boltd will enable the parties to be able to interact to activate the channel and prepare it for payments.

4. **Payment**: This can occur many times. Boltd will enable the Customer to initiate an off-chain payment of D units of currency (of positive or negative value) to the Merchant. This payment maintains the total channel balance, but updates the Customer and Merchant's ownership of the balances. The merchant does not learn which Customer or Channel was involved in the payment. This produces updated state at each party.

5. **Channel Closure**: At the conclusion of a channel interaction, the customer or merchant may initiate the closure of the channel. Boltd will enforce a time lock period (configurable) to allow either party to log a dispute. If either party dispute the balance of the channel, each party transmits a "closure token" to the Zcash network. It is expected that Zcash will include logic (via scripting opcodes) to evaluate the tokens to determine the correct balances (A, B) to pay out to the Customer and Merchant respectively. The end result will be each party withdrawing their shares of the resulting channel.

Background and Qualifications

The team consists primarily of J. Ayo Akinyele (Johns Hopkins University and YeleTech Security)

In terms of qualifications, a majority of my work is cryptographic engineering related and I've contributed to a number of open source projects as a result. In particular, I am the main developer behind the following open source projects:

- **Charm-Crypto**: a rapid prototyping framework for advanced cryptosystems. Written in Python/C and used extensively by academic researchers and practicioners around the world. See https://github.com/jhuisi/charm.

- **OpenABE**: a new commercial-grade open source attribute-based encryption library. Written in C/C++ and will be publicly available at https://github.com/zeutro/openabe.

- **Libbolt**: work-in-progress implementation of the BOLT payment channel protocol in Rust. Will be released as open source in the near future on GitHub (via Zcash Foundation).

Evaluation Plan

I anticipate three milestones that mirror the technical approach described earlier:

1. Explore optimizations for the range proofs in the Pay protocol to make it more efficient than naive solution.
2. Explore mechanisms to safely and securely link Rust into Golang code (for the purposes of integration with boltd).
3. Implement the dedicated daemon (boltd) in Go that implements BOLT communications with remote parties. This daemon uses HTTPS/JSON communications, incorporates libbolt and interfaces directly with the cryptocurrency node.
4. Investigate and document specification for all changes/improvements required in Zcash to support BOLT. This specification will be available for review by Zcash engineers and will include documenting necessary scripting opcodes, and other features from the original Zerocash protocol.

Security Considerations

The security implications of this project is to mitigate the bottleneck of on-chain transactions on the Zcash network. With off-chain transactions, the BOLT protocol dramatically reduces the transaction volume arriving at the Zcash blockchain without adding new trusted centralized entities.

Schedule

I anticpate a 6-month timeline for this project with the following milestones:

Milestone 1: Detailed specification for the potential changes to Zcash to support BOLT.

Milestone 2: Explore optimization choices for the range proofs with assistance from Ian Miers. Will deliver an implementation of selected techniques and measure performance improvements in the Pay protocol. Explore multiple approaches from Rust to Go for libbolt and will pick one of the proven methods to enable calling libbolt routines from Golang.

Milestone 3: Proof of concept boltd daemon in Go integrated with libbolt that implements BOLT communications with remote parties. The daemon will support the full functionality described in the initial libbolt specification (will be released to the community soon).

Milestone 4: Demonstrate boltd working on a test network similar to the approach taken by the lightning network project in terms of a Docker container for local testing but also deploy a public one for potential experimentation by others. In addition, will include a command-line interface to the bolt daemon (via RPC) for ease of development, testing and measurement.

Milestone 5: Demonstrate a proof of concept by compiling the bolt client into Javascript via WebASM and link to a browser plugin to support micropayment use cases. The proof of concept will not be geared for browsers running on iOS/Android mobile platforms due to technical limitations in achieving BOLT's privacy guarantees (pointed out by Ian Miers).

Budget and Justification

I am estimating a budget of **$30k** to support the continued development of libbolt in Rust, porting to Go and the implementation of boltd to support integration with the Zcash blockchain (via a test network deployment). This budget reflects compensation for my effort over a four month period dedicated to this project and cloud computing costs (e.g., AWS) for hosting a public test network for BOLT development and experimentation.

Contact Info

I can be reached at jakinye3@jhu.edu or ayo@yeletech.org.