HushList Protocol Specification
Version

David Mercer[†]
Duke Leto[†]

December 22, 2017

Abstract.

HushList is a protocol for mailing lists using the encrypted memo field of the **Zcash** protocol. It supports anonymous and pseudonymous senders, receivers and Hushlist creators, as well as public and private lists. The HushList protocol can run on any fork of **Zcash** that has a compatible memo field, though certain advanced features might not be fully supported on all chains. HushList is developed and tested on the Hush and Zcash mainnets as well as testnets (TUSH and TAZ), next to be tested is Komodo (KMD).

In addition to the above properties, **HushList** provides users with censorship-resistant storage and retrieval, since every **Hush** full node will have an encrypted copy of every **HushList** memo. Furthermore, sending and receiving via one or more blockchains is a serious deviation from traditional server-client design which easily allows a Man-In-The-Middle Attack and Deep Packet Inspection (DPI). Network traffic monitoring and correlation is made much harder, because there is no longer a packet with a timestamp and "selectors" going from one unique IP to another unique IP via a very predictable network route.

Zcash is an implementation of the *Decentralized Anonymous Payment* scheme **Zerocash**, with security fixes and adjustments to terminology, functionality and performance. It bridges the existing ***transparent*** payment scheme used by **Bitcoin** with a ***shielded*** payment scheme secured by zero-knowledge succinct non-interactive arguments of knowledge (*zk-SNARKs*).

Hush is a fork of the **Zcash** codebase (1.0.9) which generated it's own genesis block and uses the Zcash Sprout proving key.

This specification defines the **HushList** communication protocol and explains how it builds on the foundation of **Zcash** and **Bitcoin**.

Keywords: anonymity, freedom of speech, cryptographic protocols, electronic commerce and payment, financial privacy, proof of work, zero knowledge.

Contents

[†] Hush Core Developers

Introduction

HushList is a protocol for anonymous mailing lists using the encrypted memo field of the zcash protocol.

Technical terms for concepts that play an important role in **HushList** are written in *slanted text*. ***Italics*** are used for emphasis and for references between sections of the document.

The key words **MUST**, **MUST NOT**, **SHOULD**, and **SHOULD NOT** in this document are to be interpreted as described in [RFC-2119] when they appear in **ALL CAPS**. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

This specification is structured as follows:

- Notation — definitions of notation used throughout the document;
- Concepts — the principal abstractions needed to understand the protocol;
- Abstract Protocol — a high-level description of the protocol in terms of ideal cryptographic components;
- Concrete Protocol — how the functions and encodings of the abstract protocol are instantiated;
- Implications

High-level Overview

The following overview is intended to give a concise summary of the ideas behind the protocol, for an audience already familiar with *block chain*-based cryptocurrencies such as **Bitcoin** or **Zcash**.

XXX

Value in **Hush** is either *transparent* or *shielded*. Transfers of *transparent* value work essentially as in **Bitcoin** and have the same privacy properties. *Shielded* value is carried by *notes*, which specify an amount and a *paying key*. The *paying key* is part of a *payment address*, which is a destination to which *notes* can be sent. As in **Bitcoin**, this is associated with a private key that can be used to spend *notes* sent to the address; in **Hush** this is called a *spending key*.

A *transaction* can contain *transparent* inputs, outputs, and scripts, which all work as in **Bitcoin** [Bitcoin-Protocol]. It also contains a sequence of zero or more *JoinSplit descriptions*. Each of these describes a *JoinSplit transfer* which takes in a *transparent* value and up to two input *notes*, and produces a *transparent* value and up to two output *notes*.

Each **HushListMUST** have a default blockchain that it is attached to, and the default chain **SHOULD** be HUSH. The user **MUST** be able to set their GLOBAL default chain (not implemented yet) as well as a default chain for each list.

Each list also has a tadd+zaddr dedicated to that list, so the user has dedicated addresses to send psuedo/anon messages, as well as default fee and amount. The default amount is 0.0 and the default fee is currently 0.0001 but these numbers are subject to change.

HushList supports file attachments and embedding arbitrary binary data, it is not limited to ASCII.

Account Funding

On first run, **HushList** creates a new shielded zaddress z_F to fund transparent addresses for pseudonymous sending.

It may be funded by the user from any taddr or zaddr with no loss of privacy.

For each pseudonym the user sends from (may be globally used or per-list), a taddr t_L is created and a de-shielding transaction is done from $z_F \rightarrow t_L$ which will allow the user to send memos to the given **HushList** on behalf of the t_L pseudonym. Since **HushList** memos have, by default, an amount of 0.0.

For each **HushList** the user wants to be part of, **HushListMUST** create a brand new zaddress z_L (it **MUST NOT** reuse an existing address) and fund that address via a shielded $z \rightarrow z$ transaction between $z_F \rightarrow z_L$.

If there are no taddr or zaddr funds in the entire wallet, **HushListSHOULD** present the user a taddr + zaddr which can be used to "top up" the current **HushList** wallet from another wallet/exchange/etc.

HushList Contacts

HushList maintains a database of contacts which use the address as the unique ID and additional metadata. Since **HushList** supports multiple blockchains, it **MUST** have a contact database for each chain. Each chain **MUST** have it's own contact namespace, so you can have Bob on Hush and Bob and Zcash and they will not conflict.

List Creation

...

List Subscription

When the private key for a list is imported into HushList, either from the blockchain, URI or manual entry, the private key is added to the user's wallet, along with a user entered or approved name and description for the list (if provided in on-chain or uri encoded metadata). HushList creates a unique taddr + zaddr for each list so that the user may choose to send each message to the list psuedonymously or anonymously or a mixture of both.

Sending To A List

One may send to a **HushList** from a taddr (pen name, psuedonym) or zaddr (anonymous shielded address) which is implemented in the client via the z_sendmany RPC command. Up to 54 recepients may be in a single shielded transaction. v1 of HushList only supports HushLists of this size, but v2 may implement larger HushLists.

Receiving Messages

At any time later, after the transaction has entered the blockchain, memos sent to a given address can be downloaded and viewed by those parties who have valid private keys or viewing keys.

The client will poll the local full node periodically at a user specifiable default interval the same as the average block time for the chain in question. For the **Hush** chain, this is 2.5 minutes.

Costs

Sending **HushList** memos requires making a financial transaction and by default, **HushList** sends the recipient a transaction for 0.0 **HUSH** (or **ZEC** etc) with the default network fee (currently 0.0001 for **ZEC** +**HUSH**). The fee amount **MUST** be configurable by the user. In the reference implementation of **HushList** it be changed via the HUSHLIST_FEE environment variable.

References

[Bitcoin-Protocol] *Protocol documentation — Bitcoin Wiki*. URL: `https://en.bitcoin.it/wiki/Protocol_documentation` (visited on 2016-10-02) (↑ p 3).

[RFC-2119] Scott Bradner. *Request for Comments 7693: Key words for use in RFCs to Indicate Requirement Levels*. Internet Engineering Task Force (IETF). March 1997. URL: `https://tools.ietf.org/html/rfc2119` (visited on 2016-09-14) (↑ p 3).