Carrot: An Open Protocol for Naming Bitcoin Addresses

Gregg Zigler, Rick Seeger

Abstract. Person-to-person bitcoin payments currently require either in-person scanning of QR codes or out-of-band communication of 34-character bitcoin addresses. Previous attempts at improving that user experience involve trusted third parties who have custody of private keys, or who require new identifiers for users, or both. We propose a solution that minimizes trust in third parties and that maximizes use of existing identifiers such as email addresses, phone numbers, and social network names. The proposal rests on the hypothesis that many people will exchange some privacy for significant improvements in ease of use. The proposed protocol distributes bitcoin address naming across multiple participants to encourage decentralization.

1. Introduction

Outside of the bitcoin ecosystem, person-to-person (P2P) payment systems commonly use pre-existing, externally-issued identifiers such as email addresses to identify recipients of payments. Software developers have access to countless libraries that incorporate externally-issued identifiers into their applications.

Unlike email addresses and phone numbers, bitcoin addresses represented as 34-character, base58-encoded strings offer humans little hope for memorization. They lack embedded natural language words that are common in email addresses and social network names, and they lack meaningful groupings of digits found in national telephone numbering systems. What is needed is the ability to make a bitcoin P2P payment by addressing the payment to the recipient's email address or phone number.

In this paper, we propose an open protocol with three classes of participants, a query protocol with incentives for all three classes, and an extensible message format. We follow the protocol description with the incentive, privacy, and security models that underly the protocol.

2. Participants

The Carrot protocol connects users (payers and payees) with three classes of protocol participants. Classes are ordered by amount of trust required by the payer.

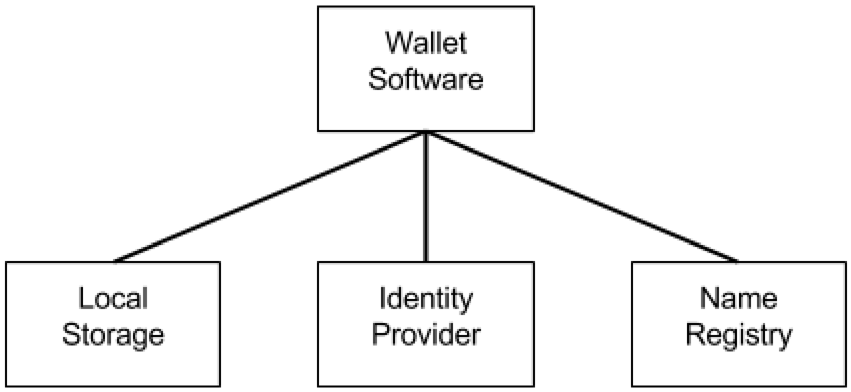
Class 1: Bitcoin wallets

Wallets use local storage to read and write confirmed mappings between email addresses and bitcoin addresses. Mobile wallet applications can also access contact lists with phone numbers and email addresses of potential payment recipients.

Class 2: Identity providers

Email service providers, mobile carriers, and social networks issue and maintain a unique identifier for each of their customers. Identity providers also provide native channels of communication with their customers. For example, mobile carriers natively communicate with their customers via SMS.

Class 3: Name registries
A name registry is any service that provides create, read, update, and delete (CRUD) functionality for mappings between externally-issued identifiers and bitcoin addresses. The protocol for these CRUD operations is specified below.



3. Registry Protocol
Users (payees) who wish to receive payments first generate a bitcoin address which they will associated with their email address with class-2 and class-3 participants. The user then registers that bitcoin address with the class-2 participant who issues the identifier of the user. For example, the email address `rick@gmail.com` would be registered with Google with the following percent-encoded API call:

```
POST https://carrot.gmail.com/rick%40gmail.com?address=1MsJbRTynV8…
```

Participants may provide additional channels for registering a bitcoin address, such as websites. Participants must confirm that the user controls the identifier using the native communication channel of that identifier. For example, to confirm ownership of an email address, the participant must send the user a confirmation email requiring the user to select a link in the email body. At any time, a user may `PUT` a different bitcoin address or `DELETE` the association.

If the class-2 issuing provider does not support the Carrot protocol, the user may register with any class-3 participant.

4. Query Protocol
Users (payers) who wish to send payments open their bitcoin wallet application, go to the send payment form, and enter the email address of the recipient. The wallet software executes the following operations.

Step 1: Query local storage
The wallet software looks in local storage for a mapping between from the email address to a bitcoin address. Previous executions of this query protocol will have populated local storage with confirmed mappings between email addresses and bitcoin addresses in step 4.

Step 2: Query the class-2 participant
If step 1 fails to find a match, the wallet software continues to step 2. There should be exactly one class-2 participant that maintains the identifier entered by the user. To find the bitcoin address of `rick@gmail.com`, the wallet software would make the following API call:

```
GET https://carrot.gmail.com/rick%40gmail.com
```

If Google finds a record for Rick's email address, Google would respond with the associated bitcoin address.

Step 3: Query class-3 participants
If step 2 fails to find a match, the wallet software continues to step 3. For example, to find Rick's email address at the kar.yt name registry, the wallet sofware would make the following API call:

```
GET https://kar.yt/rick%40gmail.com
```

The wallet software should allow the user to configure one or more class-3 participants. Precedence of multiple class-3 participants is not defined by this protocol.

Step 4: Update local storage
If the wallet software finds a match at either a class-2 or class-3 participant, the wallet software should prompt the user to confirm the bitcoin address with the recipient (payee) using an out-of-band channel, such a verbal confirmation, and then persist the address in local storage.

5. Message Format

Schema
For a given external identifier plus currency code plus network, there should be at most one address returned by the service using the following `Address` schema:

```
{
    wallet_name: string, external identifier e.g. email address
    currency: string, ISO 4217, POST/PUT default is 'XBT',
    network: string, 'btc_livenet' (POST default) or 'btc_testnet',
    wallet_address: string, address on network, no default
}
```

Example:

```
{
    wallet_name: 'rick@gmail.com',
    currency: 'XBT',
    network: 'btc_livenet',
    wallet_address: '1MsJbRTynV84oeEns3V7A3RUrTs2xcJ5Uxa'
}
```

This `Address` schema extends the Netki address format [3] in two ways. First, it adds a `network` property to support livenet and testnet addresses for a single email address. Second, it specifies ISO 4217 currency codes for the `currency` property; for compatibility, this protocol treats the Netki currency code `'btc'` as an alias for `'XBT'`.

Request
Following the BIP14 specification [1], wallet software should send software name and version, separated by a colon and enclosed in forward slashes, in the user-agent header. For example, the Satoshi client might send the following header:

```
user-agent: /Satoshi:0.9.1/
```

The entire URI, including any @-sign in the query parameters, must be percent-encoded. The the `currency` and `network` properties may be uppercase, lowercase, or mixed case.

Callers of the `GET` and `DELETE` operations must specify exactly one `wallet_name` query parameter per call, and may include `currency` and `network` parameters. Callers of the `POST` and `PUT` operations must specify exactly one `Address` per call.

Response
The response to a successful `GET` request contains an array of `Address` objects. Successful `POST` or `PUT` requests return a single `Address` object in the message body. Successful `DELETE` requests have no response message body. A `POST` request for an existing address behaves exactly like a `PUT` request.

The response header includes `cache-control: no cache`, because the user may change the bitcoin address at any time.

6. Incentives
We hope to attract a large number of participants to support the Carrot protocol. To that end, Carrot provides incentives to each type of participant.

Ease of use

Developers of class-1 wallet software can use existing contact libraries to offer familiar patterns of interaction, such as look-ahead address completion, invitation of friends, and synchronization with registries.

Financial data
Existing class-2 issuers of identifiers can gain access to financial data of their customers.

Value-added services
New class-3 businesses can attract customers by offering services such as notification of payments by email and SMS.

7. Privacy

The Carrot protocol reflects our hypothesis that many people will exchange privacy (in the form of bitcoin addresses) for ease of use. This hypothesis extrapolates decades of experience with user tolerance for web browser cookies into the world of bitcoin wallets.

Payees using the Carrot protocol must be willing to make one bitcoin address publicly visible and publicly associated with their email address, so that their friends can more easily discover their bitcoin address and send them money. Just as some users surf the web using browser privacy-mode, some payees will want separate non-public wallets for private shopping to avoid merges [2].

Meiklejohn and others [4] found that bitcoin addresses already have limited privacy due to idioms of combining change addresses, and due to clustered usage of gaming-related addresses. We therefore believe that Carrot does not represent a significant loss of privacy relative to current usage.

Some have proposed sharing extended public keys with a class-2 and class-3 participant, who would generate a new bitcoin address for each `GET` query. One problem with that proposal is that a DDOS attack might generate thousands of addresses for a single email address, which would be impossible for a mobile wallet to monitor. Another problem is that services like Gmail and Lavabit wouldn't be able to keep the extended public key both secret and on-line, due to hacking or coercion; at best, that proposal would buy temporary privacy at the cost of much greater complexity.

8. Security

To reduce the amount of trust in third parties required of payers, wallet software must do the following:

a. inform the user which participant provided the address data
b. inform the user when an address has changed
c. post new receiving addresses to a class-2 or class-3 participant

d. support a user-configurable blacklist of class-2 and class-3 participants that are known to be untrustworthy

To ensure that payees who attempt to register a bitcoin address actually control the externally issued identifier associated with it, class-2 and class-3 participants must require confirmation via the native channel of that identifier. For example, to update a bitcoin address associated with a social network handle, that social network provider must require the user to login and confirm the change. Until the user confirms the change, the participant responds to `GET` requests using data from the previous state.

9. Conclusion

We have proposed a system for payers to discover bitcoin addresses of payees via publicly queryable mappings to payee email addresses or other familiar, externally issued identifiers. The query protocol gives precedence to sources requiring the lowest level of trust by the payer, starting with addresses from local storage that were previously verified by the payer. Participants ensure the validity of the mapping by requiring users to confirm control of identifiers via familiar, native channel confirmation workflows. Users share bitcoin address from wallets whose transactions are publicly associated with their email addresses; users keep separate wallets for transactions requiring more privacy. Wallet software developers provide a familiar user experience that integrate existing contact list address books into P2P bitcoin payments.

References

[1] A. Taaki, P. Strateman, "BIP Protocol Version and User Agent", https://en.bitcoin.it/wiki/BIP_0014, 2010
[2] M. Hearn, "Merge Avoidance", https://medium.com/@octskyward/merge-avoidance-7f95a386692f, 2013
[3] Netki Wallet Name Service source code, https://github.com/netkicorp/wns-api-server
[4] S. Meiklejohn, et al. "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names," https://cseweb.ucsd.edu/~smeiklejohn/files/imc13.pdf, 2013

Last update: Sunday, July 12, 2015 7:00 PM