Bern University
of Applied Sciences

Informatikseminar

Bitmessage – Communication Without Metadata

Author Christian Basler
Tutor Kai Brünnler

Date April 26, 2015

Contents

1. Synopsis

TODO

2. Basics

While encryption technology like PGP or S/MIME provides a secure way to protect content from prying eyes, we lerned from Edward Snowden that metadata - information about who communicates with whom - is equally interesting and much easier to analyze.

With e-mail, we can only prevent this by encrypting the connection to the server as well as between servers. Therefore we can only hope that both our and the recipient's e-mail provider are both trustworthy and competent.

With Bitmessage we send a message to a sufficiently large number of participants, with the intended recipient among them. The message is encrypted such as only the person in possesion of the private key can decrypt it. All participants try to do this in order to find their messages.

3. Protocol

3.1. Nomenclature

There are a few terms that are easily mixed up. Here's a list of the most confusing ones:

message	is sent from one node to another, i.e. to announce new objects or to initialize the network connection.
msg	is the object payload containing the actual message written by a user. The term 'message' is never used to describe information exchange between users in this document.
payload	There are two kinds of payload: message payload for message types, e.g. containing inventory vectors, and object payload, which is distributed throughout the network.
object	is a kind of message whose payload is distributed among all nodes. Somtimes just the payload is meant.

3.2. Process Flow

The newly started node **A** connects to a random node **B** from its node registry and sends a *version* message, announcing the latest supported protocol version. If **B** accepts the version[1], it responds with a *verack* message, followed by a *version* message announcing its own latest supported protocol version. Node **A** then decides whether it supports **B**'s version and sends its *verack* mesage.

If both nodes accept the connection, they both send an *addr* message containing up to 1000 of its known nodes, followed by one or more *inv* messages announcing all valid objects they are aware of. They then send *getobject* request for all objects still missing from their inventory.

Getobject requests are answered by *object* messages containing the requested objects.

If a user writes a new mail on node **A**, it is offered via *inv* to up to eight connected nodes. They will get the object and distribute it to up to eight of their connections, and so on.

3.3. Messages

The messages and binary format are very well discribed in the Bitmessage wiki [1], the message description is therefore narrowed down to a description of what they do and when they're used.

3.3.1. version / verack

A *version* message contains the latest protocol version supported by a node, as well as the streams it is interested in and which features it supports. If the other node accepts, it acknowledges with a *verack* message. The connection is initialized when both nodes sent a *verack* message.

3.3.2. addr

Contains up to 1000 known nodes with their IP addresses, ports, streams and supported features.

3.3.3. inv

One *inv* message contains the hashes of up to 50000 valid objects. If your inventory is larger, several messages can be sent.

[1] A version is accepted by default if it is higher or equal to a nodes latest supported version. Nodes supporting experimental protocol versions might accept older versions.

3.3.4. getdata

Can request up to 50000 objects by sending their hashes.

3.3.5. object

Contains one requested object, which might be one of:

getpubkey	A request for a public key, which is needed to encrypt a message to a specific user.
pubkey	A public key. See 3.4 Addresses and Encryption
msg	
broadcast	

3.3.6. ping / pong

3.4. Addresses and Encryption

4. Issues

4.1. Scalability

TODO

5. Discussion

TODO

References

[1] Atheros and Jonathan Coe. Bitmessage wiki: Protocol specification, 2015.

Appendix

A. TODO