Integration of Namecoin in OpenBazaar

Chara Podimata*, Kostis Lolos†
*National Technical University of Athens - charapod@gmail.com
†National Technical University of Athens - lolos.kostis@gmail.com

***Abstract*—In this paper, we designed the integration of the open source project OpenBazaar with the cryptocurrency Namecoin. OpenBazaar is a project, a marketplace better to create a decentralized network for peer to peer commerce online—using Bitcoin—that has no fees and no restrictions. Namecoin, on the other hand, is a kind of cryptocurrency, the first fork of Bitcoin and its basic difference from it is that we can store data in it. With this integration, we give the opportunity at OpenBazaar users to declare a unique name for their store and we reinforce the security of transactions, due to the adding of Namecoin in the messages that the nodes exchange.**

Keywords – Namecoin, Bitcoin, OpenBazaar, Cryptography, Cryptocurrencies, Censorship

I. Introduction

The OpenBazaar, as a project, can by itself, solve two of the three problems of the Zooko triangle; decentralization and security. Namecoin helps as solve the third problem as well; human readability. The trust in the certification of every node does not depend (only) on the GUID of this node, but it also depends on the name that the user has declared; and this name cannot be changed, because it was created depending on the Namecoin of the node. Thus, we can visit the same store, with the same name in the future.

Below we give detailed explanations for the cryptocurrency Namecoin and its "father", Bitcoin, as long as for the OpenBazaar. In order to work on this project, we needed to use github, since OpenBazaar in an opensource project, which is currently being developed by many developers around the world. The technical features of this project are shown below.

II. Bitcoin

A. One more currency

Bitcoin is one more currency through which someone can aquire goods and services. However, it is different from the usual currencies. Bitcoin is an open-source and distributed payment system, a "banking system" that runs in the computers of its users.

The basis for its creation was set by Satoshi Nakamoto in 2008 [**?**], in a paper describing in theory how such a system could work.

B. Transactions with Bitcoin

C. Conclusions

III. Namecoin

A. Making a good idea better

B. Differences with Bitcoin

C. The .bit domain

IV. OpenBazaar

A. What is OpenBazaar

OpenBazaar is an opensource project for the creation of a decentralized network for peer-to-peer commerce - using Bitcoin - which does not involve any fees or restrictions. To put it simple, OpenBazaar is the child of eBay and BitTorrent.

At the time being, internet commerce is synonym to centralized services. eBay, Amazon and other big companies have restrictive policies and charge fees for the placement and the selling of items. They, also, accept only forms of payment which charge both the sellers and the buyers, such as credit cards of debit ones. They require from the users their personal information, something which may lead to information stolen or sold to others for profit or advertising. Buyers and sellers are not always free to exchange items and services, since companies and governments control all sorts of commerce.

OpenBazaar is a different approach to online commerce. It gives the power back to the users. Since nobody intervenes in the transactions of the users, there are no fees, nobody can censor a transaction and one can reveal only the information that he/she chooses to.

B. How does OpenBazaar work

Let's say you want to sell your old laptop. Using the client of OpenBazaar (a program which everyone can download), you can create a description for this item, with every detail he/she wants, just like he/she would do in any other online commerce service. The difference is that the price offered by the seller is not in euros or in dollars, but in Bitcoin. When this description is published, it is sent to the distributed p2p network which is constructed from other users of the OpenBazaar. Whoever searches for the keywords that you have used (ie laptop, electronics etc) will find the description of your laptop. Then, they can accept your price or ask for another one.

If you both agree to a price, the client creates a contract between you both with your digital signatures, and sends it to a third party called a notary. In case there is a conflict, an arbiter can come and help the transaction. These notaries and arbiters are also folks on the OpenBazaar network—could be

your neighbor or someone across the world—who the buyer and seller trust in case something goes wrong. The third party witnesses the contract and creates a multisignature Bitcoin account (multisig) that requires two of three people to agree before the Bitcoin can be released.

C. And what if something goes wrong?

What if you're buying a certain book from a seller, you pay the multisig, and they ship you the wrong one, or it was in poorer condition than advertised, or they don't even send a product at all?

This is where the third party comes in. Remember that a multisig requires two of three people to agree in order to move the Bitcoin. They control the third key to the multisig, so the funds will not move until either the buyer and seller work out an arrangement themselves, or the third party agrees with either the buyer or seller on how to deal with the transaction and funds in multisig.

How can you trust the third party to begin with? For that matter, how can you trust anyone on a network that guards users' privacy, thus allowing for pseudo-anonymity? OpenBazaar has a reputation and rating system that allows all parties to give feedback on other users. If someone attempts to scam another user, their reputation will suffer, and the same is true with third parties. When you go to purchase a product and select a third party for the transaction, you can see their reputation and ratings to find out if the rest of the community trusts them—or doesn't. Ensuring that these ratings are legitimate and the reputation system isn't being gamed is a difficult technical challenge, but we will talk about that later.

These steps may sound complicated, but the details are handled by the client itself. The goal of the creators of OpenBazaar is for buyers and sellers to have an even better experience using OpenBazaar than the old centralized platforms.

We will present now some information concerning a technique used in OpenBazaar, Reputation Pledges, which are an implementation of proof-of-burn. We will describe proof-of-burn, why it is necessary and how can one use Reputation Pledges.

D. Proof-of-burn

Proof-of-burn is a term used to describe the intentional and provable destruction of bitcoin for a particular purpose. When engaging in a proof-of-burn, funds are intentionally sent to an address that is unspendable, meaning those coins are gone forever.

Why would someone destroy bitcoin on purpose? In the past, this has primarily been used to bootstrap one cryptocurrency from another. Distribution of a new cryptocurrency can be determined by people burning their coins in exchange for the new currency, thus showing they are invested.

E. Reputation Pledges

OpenBazaar implements proof-of-burn in a different way. On the OpenBazaar network, users can choose to be pseudonymous, meaning you don't know their true identity. As such, it can sometimes be difficult to determine if they are trustworthy or should be avoided. A reputation system is important to help inform the network of which participants have acted honestly in the past, and which haven't. There are several facets to the OpenBazaar reputation system, which is still being built, and you can read about the overall system here.

One part of this system is Reputation Pledges. This means that a user has chosen to prove their commitment to their OpenBazaar identity by burning a certain amount of bitcoin. The act of burning coins shows the network that the user is committed to their identity because they've now expended resources on it, and if they incur a negative reputation then those resources will have gone to waste.

To help understand the importance of this, consider a similar example in the real world. Travelling salesmen were often treated with skepticism by the inhabitants of the towns they visited. Apart from the annoyance of their house calls, why would people be reluctant to purchase items from travelling salesmen? Two reasons. One, they cannot rely on reputation to determine if the salesman is selling quality merchandise or not; the other customers of this salesmen aren't located nearby. Two, the salesman has nothing to lose if their products turn out to be poor quality – there is no reputation damage if they leave, and since there is no brick-and-mortar store they've invested in, they can simply peddle their wares elsewhere.

You can think of it similarly to OpenBazaar users. If there's no cost to creating a new identity, or if there is no brick-and-mortar store to keep you in one place, you can simply abandon an identity once it has received negative feedback. Obviously online, you don't have a physical store, but a Reputation Pledge is a similar concept: you've invested resources that create an incentive to keep a good reputation and impose a significant cost for abandoning that reputation.

V. OpenBazaar - Namecoin Integration

A. Allow users to configure their user-friendly URL in the UI

```
<div class="form-group">
    <label for="inputNamecoin_id"
        class="col-sm-3 control-label">Namecoin id
    </label>
    <div class="col-sm-9">
        <input data-ng-model="settings.namecoin_id"
               data-ng-maxlength="39"
               data-ng-pattern="/^[a-z0-9\-]+$/"
               class="form-control"
               id="inputNamecoin_id"
               name="inputNamecoin_id"
               placeholder="Namecoin id"
        >
    </div>
</div>
```

```
def upgrade(db_path):
    with dbapi2.connect(db_path) as con:
        cur = con.cursor()

        # Use PRAGMA key to encrypt / decrypt database.
        cur.execute("PRAGMA key = 'passphrase';")

        try:
            cur.execute("ALTER TABLE settings "
                        "ADD COLUMN namecoin_id TEXT")
            print 'Upgraded'
            con.commit()
        except dbapi2.Error as e:
            print 'Exception: %s' % e
```

B. Relay Namecoin id to other nodes

```
# Validate that the namecoin id received is well formed
if not re.match(r'^[a-z0-9\-]{1,39}$', msg['namecoin_id']):
    msg['namecoin_id'] = ''

# Update nickname and namecoin id
self.transport.nickname = msg['nickname']
self.transport.namecoin_id = msg['namecoin_id']

if 'burnAmount' in msg:
    del msg['burnAmount']
if 'burnAddr' in msg:
    del msg['burnAddr']

# Update local settings
self.db.updateEntries(
    "settings",
    msg,
    {'market_id': self.transport.market_id}
)
```

```
self.send_raw(
    json.dumps({
        'type': 'hello',
        'pubkey': self.transport.pubkey,
        'uri': self.transport.uri,
        'senderGUID': self.transport.guid,
        'senderNick': self.transport.nickname,
        'senderNamecoin': self.transport.namecoin_id,
        'v': constants.VERSION
    }),
    cb
)
```

```
# Include sender information and version
data['guid'] = self.guid
data['senderGUID'] = self.transport.guid
data['uri'] = self.transport.uri
data['pubkey'] = self.transport.pubkey
data['senderNick'] = self.transport.nickname
data['senderNamecoin'] = self.transport.namecoin_id
data['v'] = constants.VERSION
```

C. Validate claimed Namecoin id from data received

```
def _on_message(self, msg):

    # here goes the application callbacks
    # we get a "clean" msg which is a dict holding whatever

    pubkey = msg.get('pubkey')
    uri = msg.get('uri')
    guid = msg.get('senderGUID')
    nickname = msg.get('senderNick', '')[:120]
    msg_type = msg.get('type')
    namecoin = msg.get('senderNamecoin')

    # Checking for malformed URIs
    if not network_util.is_valid_uri(uri):
        self.log.error('Malformed URI: %s', uri)
        return

    # Validate the claimed namecoin in DNSChain
    if not trust.is_valid_namecoin(namecoin, guid):
        msg['senderNamecoin'] = ''

    self.log.info(
        'Received message type "%s" from "%s" %s %s',
        msg_type, nickname, uri, guid
    )
    self.log.datadump('Raw message: %s',
              json.dumps(msg, ensure_ascii=False))
    self.dht.add_peer(uri, pubkey, guid, nickname)
    self.trigger_callbacks(msg['type'], msg)
```

```
def is_valid_namecoin(namecoin, guid):
    if not namecoin or not guid:
        return False

    server = DNSChainServer.Server(
        constants.DNSCHAIN_SERVER_IP, ""
    )
    _log.info("Looking up namecoin id: %s", namecoin)
    try:
        data = server.lookup("id/" + namecoin)
    except (DNSChainServer.DataNotFound,
            DNSChainServer.MalformedJSON):
        _log.info(
            'Claimed remote namecoin id not found: %s',
            namecoin
        )
        return False

    return data.get('openbazaar') == guid
```

D. Display remote namecoin id name

```
<tr data-ng-show="page.senderNamecoin">
    <td class="col-xs-3">Store URL</td>
    <td style="word-wrap:break-word">
        {{page.senderNamecoin}}
    </td>
</tr>
```

E. Test a basic complete request/response

```
class TestDNSChainLookup(unittest.TestCase):

    def setUp(self):
        self.dnschain_server = Server(
            "192.184.93.146",
            "NOTYETIMPLEMENTED"
        )
        self.test_cases = {
            "id/dionyziz": "dionyziz@gmail.com",
            "id/greg": ["contact@taoeffect.com",
                        "hi@okturtles.com"]
        }
        self.responses = {}
        for name in self.test_cases:
            self.responses[name] = \
                self.dnschain_server.lookup(name)

    def test_valid_JSON(self):
        for response in self.responses.itervalues():
            try:
                json.dumps(response)
            except:
                self.fail("Response was not valid JSON")

    def test_contains_email(self):
        for response in self.responses.itervalues():
            self.assertIn("email", response)

    def test_correct_email(self):
        for name in self.test_cases:
            if "email" not in self.responses[name]:
                self.skipTest(
                    'Response does not contain key "email"'
                )
            self.assertEqual(self.test_cases[name],
                             self.responses[name]["email"])
```

F. Make Travis run the tests

```
language: python

python:
  - "2.7"

script:
  - nosetests
```

VI. Future Work

References

[1] Satoshi Nakamoto: *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.

[2] Dionysis Zindros: *A pseudonymous trust system for a decentralized anonymous marketplace*, 2014
https://gist.github.com/dionyziz/e3b296861175e0ebea4b

[3] Don Davis: *Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML*, 2001
[4] Greg Slepak: *DNSChain + okTurtles*, April 26, 2014
[5] OpenBazaar: *A decentralized marketplace*
https://openbazaar.org/
https://blog.openbazaar.org/
https://github.com/OpenBazaar/OpenBazaar
[6] pydnschain: *Python library for looking up blockchain data via DNSChain*
https://github.com/okTurtles/pydnschain
[7] Bitcoin: *An innovative payment network and a new kind of money*
https://bitcoin.org/en/
Wikipedia: http://en.wikipedia.org/wiki/Bitcoin
[8] Namecoin: *A decentralized open source information registration and transfer system based on the Bitcoin cryptocurrency*
http://namecoin.info/
Wikipedia: http://en.wikipedia.org/wiki/Namecoin