DISCLAIMER:

This document does not meet the current format guidelines of the Graduate School at The University of Texas at Austin.

It has been published for informational use only.

The Report Committee for Daniel Gregory Krawisz
Certifies that this is the approved version of the following report:

Anonymity in Bitcoin and Bitmessage

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____
Vijay Garg

_____
William Bard

Anonymity in Bitcoin and Bitmessage

by

Daniel Gregory Krawisz, B.A.;M.A.

Report
Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2017

Dedication

Dedicated to my cat, Lemon.

Acknowledgements

I thank Adnan Aziz, my first supervisor, for putting up with me. I thank Vijay Garg and William Bard for supervising this project. I thank the people at Monetas and Stash, who supported my work on bmd, and the people at Mycelium, who supported my work on Shufflepuff. I thank my mother, Jane Kennedy, for her help and support.

Abstract

Anonymity in Bitcoin and Bitmessage

Daniel Gregory Krawisz, M.S.E.
The University of Texas at Austin, 2017

Supervisor: Vijay Garg

This report describes two projects created by the author which are based on ideas which originate from the Bitcoin community. The first, bmd, is a re-implementation of the Bitmessage protocol in go. Bitmessage is an anonymous and secure messaging system invented by Jonathan Warren, who was inspired by the design of Bitcoin's p2p network. [WARR1] The second is Shufflepuff, an implementation of a protocol called CoinShuffle[RUFF1] which allows several people to construct a Bitcoin transaction with an input and an output for each participant without any participant knowing who owns which output. CoinShuffle was invented by Tim Ruffing et al, and it is an upgrade of a protocol called CoinJoin, invented by Gregory Maxwell. This paper discusses the background, properties, applications, and design of bmd and Shufflepuff. There is also a report of a performance analysis on bmd.

Table of Contents

List of Tables

Chapter 1: *Fundamental Concepts*

ANONYMITY

In computer networking, *anonymity* means that it is infeasible for a passive attacker to link any two messages together as a single identity. Anonymity is a subject which is both poorly-understood and difficult to provide properly. The reason it is so difficult is that no one can hide in a vaccuum; one must hide among other people. Anonymity as a service must therefore be provided by many people to one another, all of whom wish to be anonymous. It is inherently a social phenomenon, and therefore it is inherently complicated.[DIAZ, SERJ]

The first person to publish academic papers on anonymity in a computer networking context was David Chaum[CH1, CH2]. His two classic papers are very different from one another and together they provide a good conceptual overview of anonymity.

His first paper, "The Dining Cryptographers Problem", showed that a network could provide a maximal level of anonymity with a synchronized protocol that allowed one bit to be transmitted anonymously per round. He showed that it was possible for one member in a group to broadcast a message without leaking any information about who it was. For an outside observer, the sender cannot be distinguished among n people, and for a participating observer, the sender is hidden among n-1 people (because he knows if he is the sender). This is the maximum degree of anonymity that could be achieved with n people.

Because this protocol is synchronized, it is not practical for real-world networks. This is unfortunate because the author doubts that it is possible for a non-synchronized

network to avoid leaking *some* information. Thus, for larger networks, the problem becomes one of leaking the least amount possible rather than leaking none in a way that is provable beforehand.

Thus, real-world anonymity to serve many people appears to be much more about engineering than mathematics. The general strategy of any anonymity network is to spread any information enough that no node is likely to be able to reconstruct the behavior of any identity on the network. Chaum's other paper[CH2] was about a much more practical system that could provide anonymity allowing a lot of nodes to randomly mix signals. It contained no mathematics; it simply argued that if a message is routed through a random set of a large number of servers, the message ought to be very difficult to trace. This is the basis for the Tor network, which is a large non-synchronized anonymity network, except that Tor routes IP connections rather than email only.[DING]

This paper is about two projects created by the author which are designed to provide anonymity. One of them, Shufflepuff, exists to provide anonymity in Bitcoin transactions. It uses a synchronized protocol called CoinShuffle, which means that it can provide strong guarantees of anonymity, but is limited to networks with fewer people. The second of these, bmd, implements a messaging protocol called Bitmessage[WARR1]. Bitmessage is not a synchronized protocol, so it is capable of serving a much larger network.

Proof-of-Work

Most of the ideas in this paper grew out of ideas originally proposed by the Cypherpunks, a mailing list which was active from the late 1980s to the early 2000s. The Cypherpunks were individualists who wanted to evade government oversight. David Chaum is considered to be their progenitor. Proof-of-work was an idea originally

proposed on the Cypherpunk mailing list by Adam Back.[BACK1] The idea of proof-of-work is to show an expended cost in a cryptographically secure way.

A cost function is a function that takes a cost parameter x and a message m and produces a message m' that has a verifiable expected cost c(x) to create from m. For example, suppose that H is some cryptographic hash function and m' ≅ n+m, where n is a nonce which is chosen so that H(m') is less than a given value x. This is the simplest way to implement a cost function. Because a cryptographic hash function is supposed to act like a one-way function whose output is indistinguishable from a random string, a computer that receives message m' can conclude that it could not have been produced without trying nonces until one was found that produced a valid hash value. By adjusting the cost parameter, one can make a valid nonce more or less difficult to find.

The application originally proposed for cost functions was as a spam filter. A given email address would have a certain difficulty associated with it, and any email which did not have a sufficiently difficult proof-of-work attached would be automatically discarded without being read. A sender who was not sufficiently interested in attracting the attention of the recipient to evaluate the work function would simply not send an email in the first place.

A spammer depends on being able to send large numbers of emails cheaply in order to benefit from the small fraction of people who read them. A tiny individualized per-message cost would affect him a lot more than it would affect someone who only wanted to send personalized messages.

A proof-of-work filter is an application of the handicap principle[ZA1], which is an idea that comes out of evolutionary theory. It says that a believable message must come at a verifiable cost to the sender. This idea has been tested mainly in mating

displays, though its theoretical applications are potentially much greater. There are game-theoretic models that show this to be a stable strategy under certain circumstances. [GRAF1]

Bitmessage uses proof-of-work for spam prevention. Because anonymity and confidentiality are design goals of Bitmessage, the nodes must communicate without knowing much about one another and therefore cannot automatically expect reliable information from one another. A bitmessage node must be able to limit the computational power it expends to the messages most likely to be meaningful, but it cannot look at the content of the majority of the messages. Therefore, Bitmessage nodes transmit the messages they receive based on the proof-of-work string attached to them. Only those messages worth the cost to the senders are sent into the network. The use of proof-of-work can help enable anonymous agents to cooperate by mitigating the free-rider problem.

A criticism that has been made of the proof-of-work spam filter is that such a filter would allow some spam in, if it was worthwhile to the spammer to bear the cost. [LAURIE] However, all that the filter does is put the responsibility on the sender to prove that a message is worth reading. This might well be worth it to a spammer, but the costs of sending messages would require him to consider the demographics of the addresses he has available in order to maximize the likelihood of a response. He would have to do better than to just mass email everybody. Therefore, advertisements that passed the proof-of-work filter would be more likely to be of interest to the receivers.

Spam was originally defined specifically as unsolicited email advertisements, but more recently the definition has been generalized to refer to any kind of message that a person does not want to receive. It is this more modern definition of spam that is most

relevant to a proof-of-work filter because such a filter requires the sender to be able to maximize the odds that he will produce a message to which the receiver will respond favorably.

A message that someone paid to create is more likely to be meaningful. If everyone had to compete for my attention and had to expend effort just to convince me to look at a message, then they would have an incentive to learn to produce messages that are worth reading to me. Otherwise their effort would be wasted. Therefore, if someone wishes to send an unsolicited advertisement and is willing to expend the required cost for the recipient to look at it, then it is more likely to be something that is actually worth his time. In other words, if proof-of-work does not function exactly as a more "intelligent" spam filter, then we would be better served to generalize our definition of spam to be more in accordance with the filter.

Chapter 2: *Prior Work*

COMPARISON WITH PGP

Although they are both secure messaging systems, PGP and Bitmessage have different goals in mind. An important part of the design goals of PGP is to provide a web of trust that can be used for authentication without relying on a trusted third party. As such, a person's public key may be publicly associated with him on a directory like MIT's PGP server, complete with digital signatures from his contacts to prove that the association of his key with his identity is correct. Thus, PGP does not provide anonymity or prevent traffic analysis—in fact, part of its security depends on a huge public graph of the connections of all PGP users.

An anonymous messaging protocol cannot on its own provide for authentication or it wouldn't be anonymous. However, the PGP web of trust can fruitfully be combined with Bitmessage. One can attach a public Bitmessage address to one’s public PGP key, for example. One can also use a PGP digital signature to verify one’s identity to a specific recipient over Bitmessage without revealing any information to someone watching the network.

There are considerable problems with using encryption of any kind with email. Despite the protection promised to American citizens under the 4th amendment against unwarranted searches, United States jurisprudence has evolved to establish the concept of "metadata", the contents of which may be looked at without violating the 4th amendment. For example, email headers (which include the subject line) are considered to be metadata, whereas the body of the email is not. Non-US citizens are not considered to have any rights to privacy at all by the US government and any data they send through the US is considered to be fair game for collection.

Encryption alone cannot thwart the collection of metadata, but an anonymous system can. Thus, both Bitmessage and PGP are potentially of value in a world where secrecy matters.

COMPARISON TO FREENET

Freenet is a venue for publishing documents online that is anonymous in the sense that it separates content from publisher. When browsing through Freenet, it is difficult to know which computer hosts a given document and who originally made it.[CL1, SAND1]

In Freenet, files on the network survive according to whether the nodes find them interesting enough on their own merits to be preserved. On the Bitmessage network, all messages are sent out with a specified, objective timeout period. The creator of the message can create a longer- or shorter-lived message by attaching a stronger or weaker proof-of-work to it. The creator's evident willingness to expend resources on the message is accepted by the network as an indication of the importance of the information it contains.

The fact that it is possible to encrypt any signal means that a society cannot easily act on the contents of encrypted messages and it cannot exert pressure on the specific people who send messages that are not socially acceptable. To attempt to smear an anonymity network by associating it with child pornography is nothing but a distraction. The only real question is whether we want to live in a world in which anonymity and confidentiality are available to everyone, or in a world in which they are denied to everyone.

However, it would be reasonable to suspect that people have at least one interest that they would like to keep private. If this is true, then most people could potentially benefit from an anonymity network. Since anonymity inherently requires a group of

people all to hide among one another, everyone who wants it can benefit from cooperating with one another, even if they should all find one another mutually socially unacceptable.

COMPARISON TO BITCOIN

Bitcoin[NAKA1] was the first form of digital cash to develop a value on its own in a market setting. One of the design goals of Bitcoin is to be a decentralized solution to the double-spending problem, which is an issue that must be resolved in all versions of digital cash[FINN1]. Unlike gold or physical paper bank notes, which cannot be easily reproduced, anyone can mass-produce messages on a network. A digital cash, therefore, must resolve the double-spending problem, which refers to the fact that a node in the network can send two messages which both spend the same sum of digital cash. Any distributed system which is to allow for some form of digital cash must have some means of ensuring eventually one and only one such message to be accepted as valid.

Bitcoin relies on a process called mining, in which some node attaches proof-of-work to a set of consistent transactions. The data structure which includes the sets of transactions is called a block. If there are competing blocks, the one with the higher proof-of-work is accepted as the preferred one by the network. The minimum allowable proof-of-work is adjusted to ensure that one block is accepted by the network on average every 10 minutes.

There is nothing corresponding to the double-spending problem in anonymous communication, but Bitcoin's p2p network incidentally supports anonymous mining and was therefore easily adaptable into an anonymous messaging system. Since Bitcoin is a distributed system, anyone can produce a block that will be accepted eventually by the whole network. In order to achieve the highest certainty that a given transaction will be accepted by the network, therefore, a node should want as many other nodes as possible

to know about it. In order to enable all users to participate in the creation of Bitcoin's objective history, a user must receive all the messages.

Thus, Nakamoto developed a peer-to-peer protocol to manage the distribution of messages. When a new node joins the network, other nodes send it inventory messages, which are lists of hashes of the transactions and blocks known to that node. The peer then responds with a request for everything it does not already have. Everyone on the Bitcoin network, therefore, eventually gets all the transactions that have been sent into it.

An incidental property of such a system is that of greatly reducing the information that a passive attacker could obtain from traffic analysis. Because messages tend to propagate to all nodes, it is difficult to determine which computer originally produced any given message. The Bitmessage protocol was modeled off the Bitcoin peer-to-peer protocol to serve this alternative purpose. Bitmessage is like Bitcoin without the blockchain. It tries to connect to many peers and to distribute all messages across the entire network. However, there is no need in Bitmessage for the peers to agree upon a consistent history or to remember messages after they have expired. It then becomes a protocol for transferring messages rather than for transferring value.

Chapter 3: *Justification*

USES OF BITMESSAGE

Bitmessage[WARR1] is a protocol designed to provide secure online messaging. Bitmessage messages work according to the basic principle of public-key cryptography. A public key is distributed for sending messages and a private key is kept secret for decrypting them. In practice users distribute addresses, which are hashes of public keys. The actual public key is requested and sent encrypted over the Bitmessage network.

Bitmessage also has the concept of a broadcast. Since all messages are distributed to everyone on the network, it is possible to make an address into something like a subscription service. A Bitmessage broadcast is encrypted with a private key that is generated from the address of the sender. Anyone who knows the address can decrypt the message. Since only the person who generated the address can produce a valid signature, no one who subscribes to the broadcast address can forge a message on a broadcast channel.

An interesting feature of Bitmessage broadcasts is that, because they are transmitted in a way that anonymizes both sender and recipient, it must be the case that all subscribers of a particular broadcast must receive the same message. If not, then there must be a way for a sender to treat two recipients differently, which would be impossible if they were anonymous. Therefore, to the extent that Bitmessage is successful as an anonymous messaging system, it also prevents a publisher from sending different messages to different subscribers. This is the feature that makes Bitcoin interesting as a component of Open Transactions voting pools.

DIGITAL CASH AND OPEN TRANSACTIONS

The original idea for digital cash was devised by Chaum as a means protecting a person's privacy when they spent money using a bank as an intermediary. He did not invent a new currency. His idea was for a digital bank note. It would operate like cash denominated in an existing currency. Later, ideas about what digital cash should be like were developed by Nick Szabo and others on the Cypherpunk mailing list into a form that was more like what we now know of as Bitcoin.

On the other hand, Open Transactions (OT) is a protocol which is closer to Chaum's earlier idea[ODOM1]. OT allows for digital receipts for the exchange of goods which are signed by both sender and receiver. OT has options for digital blinding, like that invented by Chaum to protect users' privacy.

An application of such a system is an untrusted bank. By relying on a set of signed receipts for all withdrawals and deposits, the bank and the user can both prove what the balance of the account should be. If the bank attempts to appropriate funds, this is provable by the user. The bank still needs to be audited in order to prove that it actually has the funds corresponding to its customers' balances.

These two ideas can be combined. OT can be used on top of Bitcoin. This allows for security to be provided for certain uses of Bitcoin to which it is not well-adapted on its own.

OPEN TRANSACTIONS VOTING POOLS

A problem which has affected Bitcoin as soon as it first began to attract investment was that although Bitcoin mostly eliminates the need for banks, for those services which still required an agent to store funds on behalf of a client, it was wildly insecure. In fact, it was much less secure than any currency which had existed previously,

and for the same reason it was so wonderfully secure under other circumstances: whoever controlled the private key had absolute control. A digital signature, and only a digital signature, was required to make a valid payment. Thus, because users could store their funds in cryptographic private keys, a user was capable of securing his funds against misappropriation to an extent that had been impossible with previous currencies, as long as that user controlled the private key.

Usually this was possible because most monetary services for which we use agents today could be done automatically with cryptography or by the Bitcoin network's peer to peer protocol. But when this was not the case, the agents on whom one had to rely could misappropriate funds more easily than any previous embezzler could with other people's dollars. Therefore, such embezzlement occurred at a rate which was laughable. [SCAM1]

Although this has improved in recent years, it has not improved as a result of a sufficient improvement in our use of cryptography, but rather because businesses have begun to achieve a sufficient value in their brand as to make the value of misappropriated funds less, not worth the requisite destruction of the brand. The consumer preference for brand name recognition therefore forces potential competitors to demonstrate a similar level of honesty.

While the economic value of honest branding is not to be dismissed, a form of security that comes with a mathematical proof is preferable if possible. Open Transactions aims to provide a cryptographic solution using what its designer has called voting pools. A voting pool consists of several computers, each of which provides some services as a Bitcoin agent, which all jointly control a single wallet containing the total of all the bitcoins they hold as agents. The wallet is multisig type, which means that a

certain minimum number of signatures is required to spend Bitcoins from it. Therefore, the numbers of the voting pool must agree, up to a certain number of them, to allow for balances to be redeemed, and none of them can act on its own to misappropriate them.

Because the Bitcoin blockchain is public, it is possible for the voting pool to prove that it controls a certain sum of Bitcoins. However, there is still a problem proving that the amount of bitcoins in the voting pool is the amount that ought to be there.

A well-known trick in accountancy is to make two sets of books, one which is shown to an auditor and another which records the real state of the finances of a company. The analogous trick with a Bitcoin bank is to send two sets of messages which give a different impression about how many Bitcoins should be in the bank. One set of messages is sent to the bank's customers, and another to the auditor. Each voting pool must be capable of cooperating with an auditing service which proves or affirms as to the correspondence between a set of transactions in the blockchain and a provably complete record of deposit and withdrawal orders authorizing the set of transactions. The provably complete part is where Bitmessage comes in.

The solution is to use a public channel that cannot be easily be censored, and this is exactly what a Bitmessage broadcast channel accomplishes. Once a message is broadcast, anyone with the key can read it, and since anyone can be a Bitmessage node, the recipients of a broadcast cannot be controlled, once the message is sent out. An auditor, therefore, can be assured to receive the same messages as those received by the customers.

If the users of the voting pool broadcast their deposit and withdrawal orders over a Bitmessage channel, then anyone listening to this channel can be certain that he has a copy of every message sent over the channel. This means that the auditor can be certain to

have seen every relevant message. Otherwise, it would be possible for a member of a voting pool to show different records to its auditor than it does to its clients when they withdraw. This would allow a member to create fake customers and make fake deposit orders that cover up real customers. Then when the customers withdrew their bitcoins, the voting pool member could make fake withdrawal requests that would funnel bitcoins into his own account.

Chapter 4: *Engineering*

THE BITMESSAGE PROTOCOL

I now discuss some of details of the Bitmessage p2p protocol. Bitmessage assumes that its nodes communicate over a channel. Most likely this would be over TCP or Tor. When two bitmessage nodes first connect, they engage in an initial handshake. The handshake consists of a version and a verack from each of them. The Bitmessage protocol has changed in minor ways in the past, so the version message is useful for checking whether and how two nodes can communicate. The verack is an empty message.

After the initial handshake, the nodes send inv and address messages to one another. These messages tell each other about the objects and IP addresses they each know about on the network. As new messages appear on the network, the nodes can update each other as to the new information.

Each node requests the objects that they do not have using the getobject message. This message is simply a list of hashes, just like inv. The node which received the getobject responds with a series of object messages.

A bitmessage object contains an int value which specifies the type of object. Any value is allowed and any type of object will traverse the network, but only four are defined in the Bitmessage protocol so far.

- msgGetPubkey: A request for a public key corresponding to a given Bitmessage address. The request is encrypted to the requestor's public key. Without some other means of interacting with a recipient, the public key must be requested in order to be received.
- msgPubkey: The public key that was requested. Can only be decrypted by the sender of the request.

- msgMsg: A Bitmessage message. Bitmessage has a strange way of using the word *message*. The whole idea of Bitmessage is that it is a messaging protocol, but it relies on a protocol which requires instances of it to send packets to one another which are also called messages, even though they may not contain the messages that users of Bitmessage send to one another. Why not call this specific kind of message a bitmessage? That is sometimes how people speak informally, as in "I'll send you an email." There is, furthermore, a concept of the formatting of the contents of the message object. It can be divided into subject and message.
- msgBroadcast: a broadcast.

BITMESSAGE AND IMAP

In the case of Bitmessage, there are potential benefits to be derived by connecting the Bitmessage network to one that is already huge, namely those people using email. If a Bitmessage node were also an IMAP server, any IMAP client would be able to interface with the Bitmessage network instantly, via the same interface by which they use email. This would greatly ease the transition to Bitmessage.

It furthermore allows the use of Bitmessage to expand into smart phones, which is important given the prevalence of mobile devices on the Internet. Those whose only connection to the Internet is through a phone cannot rely on Bitmessage without connecting to a server because an Internet connection through a phone is not cheap or fast enough to satisfy the demands of the bitmessage network.

Someone who did this would have to allow a server to manage his cryptographic keys, but he could maintain confidential communication by combining Bitmessage with PGP. This would compromise the user's anonymity to accommodate his bandwidth requirements but he would still be anonymous from the standpoint of an observer of the

Bitmessage network, and someone watching his own connection would only see that he is connecting to a standard server.

Unfortunately, the standard implementation of Bitmessage, PyBitmessage, was designed as a standalone application designed with a GUI to be used directly by a human. As important as a GUI might be, no decision made about it nor any amount of time thinking about it is likely to have consequences as fateful or prolonged as the corresponding effort upon the core protocol. Furthermore, if the program acted as an IMAP server then the GUI problem is already solved by Thunderbird and Apple Mail.

My company needs something to be used primarily by other programs rather than directly by humans. This need is the motivation to fund a reimplementation of the protocol. The most effective distribution of programmers in a large enough project would save those with the rarest skill or potential to make only the most momentous decisions. Such a distribution restricts the potential consequences of every decision according to its expected value, predisposing the good of the resulting product to predominate over the bad.

DESIGN OF *BMD*

bmd is separated into two applications, which communicate with one another over an RPC interface. One of them is a network node, and its job is to interact with the Bitmessage network and manage the set of pending objects. The other is an IMAP server, and its job is to decrypt and encrypt messages and to interact with a user through the IMAP interface.

This design allows for a separation between the cryptographic keys and the Internet connection, as well as a separation of function by intensiveness of computational resources. The network node will tend to be much more demanding of system resources

than the user interface, except for users who are constantly sending messages. Several users can have their own instance of the IMAP server that connects to a single network node. Someone who wanted to use Bitmessage for his personal communication could run the IMAP server on a phone, and someone who needed to send messages all the time could have a dedicated computer optimized for proof-of-work computations which runs the IMAP server.

Lite clients could connect to the IMAP server remotely. If the user did not own the computer running the IMAP server, this would be a security risk because he would not control his own keys. He could still use PGP to prevent the server from reading his messages, but he would still be trusting the server with information that could compromise his anonymity. This would be a reasonable compromise under certain circumstances.

BMD AND *BTCD*

Bmd was designed with the intention of re-using much of the code from btcd, a Bitcoin full node developed by Conformal. Because Bitmessage was designed from the Bitcoin protocol, there was a significant possibility of adapting existing work.

The btcd team has developed a very useful foundation that could readily be adapted to any peer-to-peer program. It manages peers and it has a well-designed abstract system. It was very easy to adapt. The team had also developed a library for ECDSA cryptographic operations, which are the same in Bitcoin and Bitmessage.

As bmd came together, it became easier to develop some of the recycled material in our own way rather than try to keep up with the btcd team. Different timing and priority constraints required us to work independently in ways that might have allowed for more adaptation otherwise.

Chapter 5: *Performance Analysis of bmd*

bmd is separated into two parts, an IMAP server and a Bitmessage network node, and they each have different performance requirements. The IMAP server was analyzed to find out how many messages a user could send and how quickly they were received. The network node was analyzed to get an idea of how much more traffic the network as a whole could support before it became prohibitive on system resources. All tests were done on a computer with 8 cores running at 2.60 GHz.

There is a limit to the number of messages and amount of data that can be sent over the network at once, due to the proof-of-work requirements on valid messages. The limit should adjust itself slowly as the network becomes more congested because people will begin to create more addresses with higher proof-of-work requirements. Therefore, the result is only meaningful in the network's current state. Several short (1500 bytes) messages were sent and the time required to run proof-of-work and for them to be received was recorded. The mean time taken to run proof-of-work was found to be 7.4 ± 4.8 seconds.

There is no need to narrow this number down any further because proof-of-work is inherently probabilistic. The number is sufficiently accurate to prove that a user who wants to use Bitmessage like his personal email can send at least one message every 10 seconds.

How long does it take a message to arrive once it is sent? To answer this question, two instances of Bitmessage were run on the same computer and messages were sent at random times over the course of a day. The Bitmessage network is continually changing as nodes come off- and online. So once again, the real answer is probabilistic but it is

possible to confirm that messages will arrive in a reasonable amount of time. The mean time for a message to be received was 250. ± 180 seconds.

Theoretically, it should not make a big difference if the sender and recipient are near to one another, or even the same computer. Bitmessage nodes connect to one another randomly and they do not attempt to optimize. Messages should generally tend to travel along faster routes, but two computers in the same location would not pass messages instantaneously unless they randomly happened to be connected to one another. It would be nice to test this with computers on opposite ends of the world.

As to the network node, the big performance issue is the amount of data that needs to be processed. The network node must handle all messages passing through the Bitmessage network. The network node has to do it fast enough and it has to have enough memory. Therefore, the network node was examined to see how quickly it would run out of time or memory.

When bmd is first turned on, it has to catch up with the rest of the network and download all the messages currently circulating in the Bitmessage network. During this time, it is working as fast as it can.

It is possible to test how fast bmd is by checking how much information it processes normally compared to the amount it processes when it is getting up-to-date with the rest of the network. While bmd is catching up with the rest of the network, it is processing messages as fast as it can until it has the same messages as the other nodes. This can be used as a baseline to determine the maximum data processing capacity of bmd. Several times over the course of a week, bmd was restarted and its database was deleted. All Bitmessage objects that it processed were logged by time and size in bytes for several hours thereafter.

When bmd was catching up with the rest of the network, it processed Bitmessage objects at a rate of 66 kb / second. After it had caught up, it processed objects at a rate of only 138 bytes / second. Clearly there is lots of room for the Bitmessage network to grow before bmd needs to be optimized further.

The memory usage of the program was observed once a minute for a 24-hour period of normal operation. During this time, there was an initial rapid rise as the program caught up with the rest of the network, whereupon the memory usage stayed within about 250 mb to 350 mb. There was a jump upward about four hours in whose cause is unknown to the author, but after that there was a very gradual downward trend. bmd's memory usage is not prohibitive for modern computers.

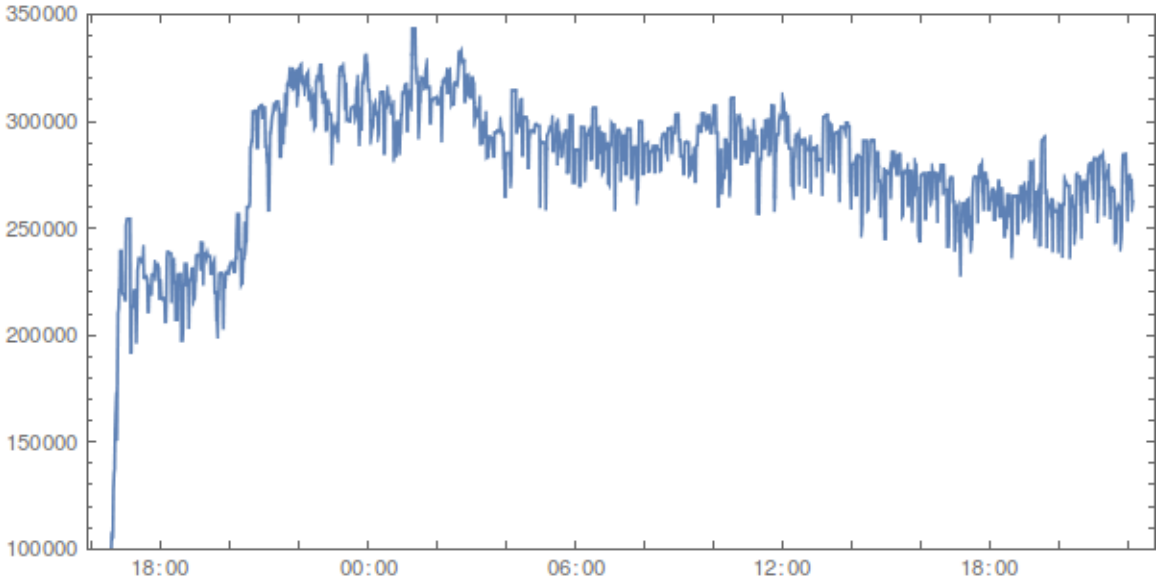

Figure 1: Memory Usage of bmd

Chapter 6: *Shufflepuff*

ANONYMITY IN BITCOIN

Although mining in Bitcoin is anonymous, spending is not. Because Bitcoin was built with the requirement that there should be no trusted party, the correctness of the blockchain needs to be publicly verifiable. Consequently, a Bitcoin transaction leaves a lot of public information from which much more than simply the validity of the blockchain can be learned.

Anonymity in Bitcoin means two different things: it means that a given transaction cannot easily be linked to an identity outside of the Bitcoin network, and that any two transactions in the blockchain cannot easily be concluded definitively to belong to the same identity or to different identities. The second problem is the only one that will be addressed in this report.

Bitcoin's blockchain can be thought of as a weighted directed acyclic graph in which transactions are nodes, and inputs and outputs are respectively the incoming and outgoing ends of the edges. The weights of the graph are the amounts in Bitcoin which are transferred along each input/output pair. A requirement of a valid Bitcoin transaction is that the sum of the outgoing weights should be less than or equal to the sum of the incoming—otherwise this would mean someone has spent more than he owns! All of this information is public, and barring the development of some amazing new zero-knowledge constructions, it is likely to stay public as Bitcoin evolves.

Because it is possible to track the movement of funds from one address to the next, it is possible to track individual identities in the blockchain. The technique, known as taint analysis, is not perfectly accurate, but it is possible to link a series of transactions

as belonging to a single identity with a high degree of confidence. Anonymity in Bitcoin is a significant challenge because the block chain must remain publicly auditable.

Unfortunately, a major problem with Bitcoin anonymity is the lack of publicly available tools for breaking it. This means that people are not very aware of how anonymous they are or what they are revealing to an attacker scanning the blockchain with the eye of a detective. Although my project does not specifically address this issue, the author believes it very important that people should be able to see what can be inferred about them from the information in the blockchain. This will make them want anonymity a lot more. Furthermore, it will make things a lot more fair, insofar as we know that there are private tools for taint analysis.

For example, suppose there were some sort of transaction which could, in principle, serve a variety of purposes but which, in practice, is understood to be produced by one software package that serves a particular purpose. Then that transaction would, in fact, reveal a more information than one might expect. Therefore, the information revealed by a given transaction is as much about its context than itself.

It is unlikely that any widespread and public taint-analytics system would match the sophistication of one used in secret by people with power and an interest in de-anonymizing Bitcoin. An organization like the National Security Administration would be capable of putting an extraordinary effort into developing the most nuanced classification of Bitcoin transactions in order to extract the most information from them.

BITCOIN TUMBLERS

There is most likely no general means of securing anonymity in Bitcoin. Instead, Bitcoin anonymity is best addressed by following several approaches. The two basic approaches to Bitcoin anonymity are tumblers and join transactions.

Tumblers and join transactions are to one another like Chaum's original email routing protocols and the dining cryptographers. A tumbler mixes many people over an extended period of time whereas join transactions are instantaneous from the standpoint of a blockchain observer.

The way a tumbler works is that someone sends his bitcoins to the tumbler. The tumbler holds his bitcoins for a period of time and receives bitcoins from many other customers. The coins are then gradually released such that the transactions by which they entered are disconnected from those by which they leave. The user's coins have effectively been mixed with all those that have gone through the tumbler while they were in the tumbler's possession.

The disadvantage of a tumbler is that the tumbler controls its customers' funds for a period of time. This means that the tumbler can perpetrate an exit scam and disappear with all the money.

JOIN TRANSACTIONS IN BITCOIN

Another possibility is the use of join transactions. A join transaction is created by N Bitcoin users who construct a single transaction with inputs and outputs for each of them. The transaction is constructed so that a blockchain observer cannot determine which of the outputs corresponds to any given input.

An advantage of join transactions over tumblers is that each participant maintains complete control over his bitcoins. A stringent constraint on creation of a successful join transaction is that all participants must spend the same number of coins. If this were not the case, it would be possible to connect inputs to outputs in the join transaction by comparing the amounts.

A more flexible join transaction is possible if participants are allowed to send coins to one another at the same time. Because a blockchain observer does not know who transferred funds to whom in the join transaction, it should be possible for any input to correspond to any output as long as the people corresponding to those inputs have transferred funds, or if there is a connected sequence of transferred funds between people which includes those two. The author does not know of this possibility having been explored in much detail.

DESCRIPTION OF COINSHUFFLE

CoinJoin was the first join transaction to be described in detail. Secure protocols for card-shuffling have long been a part of cryptographic lore. CoinJoin is essentially a secure card-shuffling protocol with Bitcoin addresses instead of cards.

CoinJoin begins with a set of N users wishing to construct a join transaction. The users are numbered from 1 to N. Each of them has a public encryption and private decryption key to use in the protocol. Player 1 begins by successively encrypting his new address in which his anonymous coins are to be deposited with the keys of players 2 to N in decreasing order. He then passes this information to player 2. Player 2 can decrypt first address with his key, but it is still encrypted with the keys of players 3 to N. He then encrypts his new address with the keys of players 3 to N and passes both addresses, randomly ordered, to player 3. Players 3 through N follow the same procedure, each adding in their own address and reshuffling until the final player ends up with the final list of addresses. The list of addresses will be randomly ordered, and no player knows which address belongs to any other player.

A later invention is called CoinShuffle, and this is the protocol that Shufflepuff implements. CoinShuffle improves upon CoinJoin by making its users provide enough

information to one another to enable them to identify and exclude malicious participants. A CoinJoin player can disrupt the protocol in a way that prevents users from identifying the miscreant, which means that the group as a whole can no longer work together to produce a join transaction. This is a serious problem because it allows trolls to disrupt the entire process.

CoinShuffle resolves this issue by defining an elaborate blame phase to the protocol which begins once any problem is detected. Players then send earlier messages to one another, enabling them all to determine who misbehaved. They can then exclude the troll and start over.

CoinShuffle assumes that N people are already in communication with one another and are ready to produce a join transaction. Their ability to find one another and agree to begin the protocol is assumed, but it is actually a difficult problem in its own right.

THE TAKER/MAKER MODEL

There are two other projects known to the author which are able to construct join transactions. One of them is called Dark Wallet, and the other is JoinMarket. Both of them rely on what is called a taker/maker model to collect participants for a join transaction.

In the taker/maker model, one participant acts as the taker and others act as makers. The taker signals that he wants to make a join transaction for a given amount and waits until enough makers sign on to it, whereupon the protocol begins. The taker may have to pay the makers to construct the join transaction with him.

In the taker/maker model, the takers pay the makers for being immediately available to make a join transaction. They are not paying the makers for successfully

providing anonymity. A maker could well construct the transaction and then immediately destroy all anonymity by merging his output with another one that isn't anonymous. Someone who wishes to construct a join transaction has the need to determine whether the people he wishes to join with will provide for his anonymity, which is a more fundamental problem than whether they are immediately available to do so.

Furthermore, the taker/maker model is only of use to people who have an immediate need to make join transactions. There is no reason that people should have this need in general, and therefore there is no reason that anyone should necessarily want to be either a taker or a maker. The primary concern is the provision of anonymity itself, and this concern should be addressed before worrying about providing it within a narrow time frame.

THEORETICAL CONTRIBUTIONS

I believe that the taker/maker model solves a less fundamental problem before more fundamental problems have been sufficiently addressed. A concern which, in my opinion, is mismanaged and which has hampered the development of good join transaction platforms until now is a desire for all parties to be anonymous throughout the entire process. This is not necessary. It is much more important to ensure that the information in the blockchain is obscure than to ensure that all information everywhere is obscure. The information in the blockchain is permanent and public. Thus, even if all participants are not anonymous in an absolute sense, they are doing a lot to protect themselves simply by keeping information out of the block chain.

Insofar as an anonymous peer can avoid consequences for bad behavior, it is very difficult for anonymous peers to cooperate with one another. A set of anonymous peers attempting to form a group so as to construct a join transaction should be able to provide

some assurance to one another that they are serious about constructing it. This can be done in two ways that the author is aware of: if the peers wish to remain anonymous, then they should be able to prove that they are invested (handicap principle); otherwise, they can choose not to be anonymous to one another and show that they have a history of successfully constructing join transactions.

There is no reason that people who are anonymous from the standpoint of a blockchain observer should necessarily be anonymous to one another. Therefore, the author proposed a design in which a server acts as an open platform in which people meet and register to participate in join transactions which are scheduled to occur at some time in the future.

The server can provide the service of assuring peers that they can reasonably expect most of the people they meet through the platform will treat one another responsibly. It can do this in two ways: by observing a player's previous behavior or by collecting fees. If a player is able to prove to a server that he has participated in previous join transactions without having compromised his anonymity or those of the other players, then that is a sign that he will continue to be responsible in the future. If a player pays a fee to take part in a join, then that shows they are invested and therefore likely to be responsible. (This is another application of the Handicap Principle).

Some combination of these two options is able to provide a reasonable assurance to the players in a join protocol that most of them are most likely going to behave responsibly. There could even be different levels of assurance to provide for people with different levels of paranoia.

Chapter 7: *Artificial Intelligence and Privacy*

The more people you have experience with, the more finely they can be classified and the more nuanced can our reactions be to them. If you are a salesman, for example, you will gradually learn many reasons that a person might be interested in your product and tailor your pitch to address those specific reasons depending on the type of customer. The same idea applies if you are a con artist, propagandist, or provocateur.

This principle extends beyond our own human intelligences. As the number of people within a data set increases, more correlations can be found among them. Finding these correlations is not necessarily an easy problem, but it is one for which heuristic solutions can be generated automatically. In other words, it is a problem for artificial intelligence.

Large organizations that interact with many people at a time therefore have the incentive to develop an individualized interaction for each customer so as to maximize the value of the interaction. Such an organization is capable of personalizing its interactions to a degree far beyond what is possible for ordinary humans.

The application of artificial intelligence to the problem means that these correlations will become known and acted upon automatically, without necessarily having to be reviewed by a human. Ultimately the organization will be able to extract all possible meaning from its interactions.

We can expect, therefore, that any information we reveal to large organizations reveals much more about ourselves than we know. The organization has the knowledge which makes it meaningful, and we do not. In effect, this gives these organizations the ability to manipulate people en masse by tailoring their experiences to their individual vulnerabilities.

It is known, for example, that Facebook has conducted experiments upon its users which show that it can cause people to become depressed or affect the outcome of elections.[KRAM, BOND] People willingly allow Facebook to present them information filtered by its algorithm, and it is quite difficult to know whether such an experiment has been performed unless Facebook reveals it.

The problem, obviously, is much bigger than exploitation by companies. Companies just want your money but governments want a lot more, and they have access to enormous amounts of data on people's behavior as well. The government of China has developed a game that encourages people to interact with it.[OSB] The game also rewards people for being good citizens, but to me the really sinister aspect of it is the fact that the constant interaction is the perfect form of data collection.

Someone who believes he has nothing to hide cannot test this belief because he does not know what his information means to those to whom he reveals it. He might well have something that he would want to hide! In fact, he very likely does because he can be almost certain that he is being watched by people who know a lot more than he does.

Thus, anonymity is a service of general applicability. It is not only for those who are considered nefarious by social consensus, but rather for anyone capable of being manipulated. This group naturally includes everybody. Anonymity is an essential tool for remaining autonomous in a world dominated by vast intelligences. If people were more aware of the risks of revealing their information, they would be more inclined to want anonymity for themselves.

Conclusion

The projects discussed in this report, bmd and Shufflepuff, were built by the author to provide anonymity. bmd has the potential to become an important part of the infrastructure that preserves our anonymity and free speech. It has properties which make it useful for certain purposes which cannot easily be reproduced with other secure messaging services. It has the potential to be more than merely an anonymous version of email. Shufflepuff is much more limited in its application, but it has the potential to be very useful to some Bitcoiners.

References

[BACK1] Back, A. (2002). Hashcash - A Denial of Service Counter-Measure. [pdf] Available at: http://www.hashcash.org/papers/hashcash.pdf. SHA-256: fb8b1a6a3f8cdf48b189e77c0355e0f2ffb0f9d6d51d7f1d8c2b29f61c0d33bc.

[BOND] Bond, R., Fariss, C., Jones, J., Kramer, A., Marlow, C., Settle, J., Fowler, J. (2012). A 61-million-person experiment in social influence and political mobilization. Nature, 489 (7415), 295–298.

[CH1] Chaum, D. (1988). The dining cryptographers problem: Unconditional sender and recipient untraceability. J. Cryptology, 1 (1), 65-75. [online] Available at: http://www.cise.ufl.edu/~nemo/anonymity/papers/chaum-dc.pdf. SHA-256: 538c00dc46a0d340fad90282cdeeb3f5a25d7eb7a4f8cb28406e72140cf7a357.

[CH2] Chaum, D. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, February, 24, (2), 84-90. [online] Available at: http://nakamotoinstitute.org/static/docs/untraceable-electronic-mail.pdf. SHA-256: f7996dbb8496c100a67db1f627342d0187a1f6625afc021f24610520adc7e1c6.

[CL1] Clarke, I. (1999). A Distributed Decentralised Information Storage and Retrieval System. Undergraduate. Division of Informatics, University of Edinburgh. [online] Available at: https://freenetproject.org/assets/papers/ddisrs.pdf. SHA-

256:
bf0881a19f165af74808c87b5a4f70d7d9cb5eb8fb1ddf8e45769ff7f68a4b75.

[DIAZ] Diaz, C., Stefaan, S., Claessens, J., Preneel, B. (2002). Towards Measuring Anonymity. In: Proceedings of PET 2002 [online] San Francisco: Semantic Scholar Website. Available at: https://pdfs.semanticscholar.org/8db1/4b49ee49f606d4123144154f38f9406af7db.pdf. SHA-256: a2f25654ba297242b291e8bc44de1f906bdd888a58bbcce07a9ff42a8720935d.

[DING] Dingledine, R., Mathewson, N., Syverson, P. (2004). Tor: The Second Generation Onion Router. [pdf] Tor Project Website. Available at: https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf. SHA-256: 03e42dd79d92a17366ce786ddd00ddd5d6be7629331eeea062753d243930853c.

[FINN1] Finney, H. (1996). Detecting Double Spending. [online] Nakamoto Institute Website. Available at http://nakamotoinstitute.org/detecting-double-spending/. SHA-256:
fe1bfb4d2bf2ec2009421afab2650aceeff81cd1024de4c0e944663ff6618223.

[GRAF1] Grafen, A. (1990). Biological signals as handicaps. Journal of Theoretical Biology, 144 (4), 517-546.

[KRAM] Kramer, A., Guillory, J., Hancock, J. (2014). Experimental evidence of massive-scale emotional contagion through social networks. Proceedings of the National Academy of Sciences, 111 (24), 8788-8790.

[LAURIE] Laurie, B., Clayton, R. (2004). 'Proof of Work' Proves Not to Work. [pdf] Cambridge, United Kingdom: Cambridge University Computer Laboratory Website. Available at: https://www.cl.cam.ac.uk/~rnc1/proofwork.pdf SHA-256: 37e914a82c7f706b9292beea094d85f4824a7401020cca344876f6cda9d769d9.

[NAKA1] Nakamoto, S. (2008) Bitcoin: A Peer-to-peer Electronic Cash System. [pdf] bitcoin Website. Available at: https://bitcoin.org/bitcoin.pdf SHA-256: b1674191a88ec5cdd733e4240a81803105dc412d6c6708d53ab94fc248f4f553.

[ODOM1] Odom, C. (2015). Open Transactions: Secure Contracts between Untrusted Parties. [online] open transactions Website. [pdf] Available at: http://www.opentransactions.org/open-transactions.pdf. SHA-256: 21c52ceb84ee084e03edcd405264331a6636c675b592712f82f1002480d1f603.

[OSB] Osborne, S. (2015). China has made obedience to the State a game. The Independent, 22 December. [online] Available at: http://www.independent.co.uk/news/world/asia/china-has-made-obedience-to-the-state-a-game-a6783841.html

[RUFF1] Ruffing, T., Moreno-Sanchez, P., Kate, A. (2014). Coin Shuffle: Practical Decentralized Coin Mixing for Bitcoin [online] petsymposium Website. [online] Available at: https://petsymposium.org/2014/papers/Ruffing.pdf. SHA-256: 607ac3ae7891a3336fcc3049ee8bf37c4dda4a2a615eeb120cf1e48b89a9a848.

[SAND1] Sandberg, O. (2005). Searching in a Small World. PhD. Division of Mathematical Statistics, Chalmers University of Technology and Goteborg University, Goteborg, Sweden. [online] Available at https://freenetproject.org/assets/papers/lic.pdf. SHA-256 286090c52c6af7817715660390d7b46b07fb061f4bb5881e4bd0a3d9dddf02c3.

[SCAM1] dree12. (2014). List of Major Bitcoin Heists, Thefts, Hacks, Scams, and Losses. [forum post] Bitcoin Talk. Available at: https://bitcointalk.org/index.php?topic=83794.0 [Accessed 18 October 2016].

[SERJ] Serjantov, A., Danezis, G. (2003). Towards an Information Theoretic Metric for Anonymity. In: Dingledine R., Syverson P. (eds) Privacy Enhancing Technologies. PET 2002. Lecture Notes in Computer Science, vol 2482. Springer, Berlin, Heidelberg. Available at: https://www.cs.ucsb.edu/~ravenben/classes/595n-s07/papers/anon-serj.pdf. SHA-256: 6e89d41eaaf7ca2381817df3a51453742fd4eea573f8eed0e620c23acd075dba.

[WARR1] Warren, J. (2012). Bitmessage: A Peer-to-Peer Message Authentication and Delivery System. [online] bitmessage website. Available at: https://bitmessage.org/bitmessage.pdf, SHA-256: 090392868656cd87bcf08f2abe6b59832a9a4f8a9757b329957b37db2f90313e.

[ZA1] Zahavi, A., Zahavi, A. (1999). The Handicap Principle: A Missing Piece of Darwin's Puzzle. New York: Oxford University Press.