Bitcoin Transaction Anonymization Through Randomized Dynamic Transactions

Eli Hickox
ACM SIGCoin
University of Illinois at Urbana-Champaign

April 11, 2015

Contents

Abstract

This paper discusses the concept of randomized dynamic transactions as a method for transmitting Bitcoin anonymously. We will begin with defining what it means to be anonymous in a transaction graph and by defining what it means to lack anonymity in a transaction graph. This method builds upon an existing method of transaction anonymization, known as a shared wallet. A shared wallet is a bitcoin address, or collection of addresses, operated by a trusted third party which essentially acts as a middle-man for a Bitcoin transaction. Anonymization is achieved by the mixing of other users Bitcoins within said wallet. This paper discusses the potential problems one may encounter when implementing a shared wallet and seeks to address them individually.

1 Introduction

The bitcoin transaction network can be represented as a directed graph, where each node represents a public key, or address, and each directed edge represents the flow of bitcoin from one address to another. Let $\mathcal{A}$ represent the bitcoin address network. An example $\mathcal{A}$ is given below:

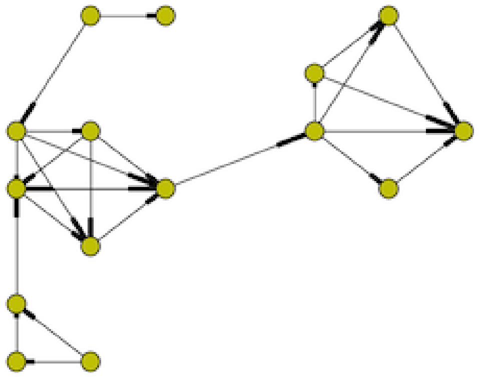

Figure 1: An example Bitcoin Address Network, $\mathcal{A}$

Now, individual addresses don't necessarily correspond to individuals in the network of bitcoin users. An individual may posses many addresses, and in the ideal use case, a new address would be generated for each transaction to preserve anonymity. However, address reuse is common among many bitcoin users who are not necessarily concerned with preserving anonymity. Because of this, it is possible to create a separate graph from $\mathcal{A}$ by associating identities with subgraphs corresponding to maximally connected components within the larger address set [4]. Let $\mathcal{I}$ represent the bitcoin identity graph.

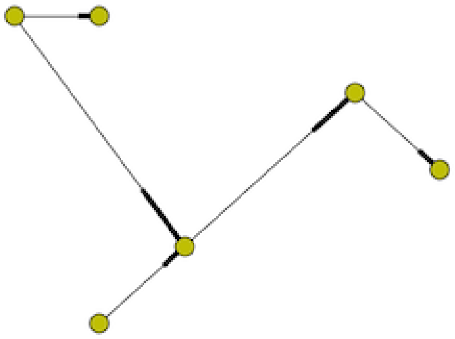

Figure 2: An identity graph, $\mathcal{I}$ constructed from $\mathcal{A}$

As you can see, since our $\mathcal{A}$ contains many directed cycles and cliques, it is easy to construct $\mathcal{I}$. Now consider another bitcoin address network, $\mathcal{A}_2$.

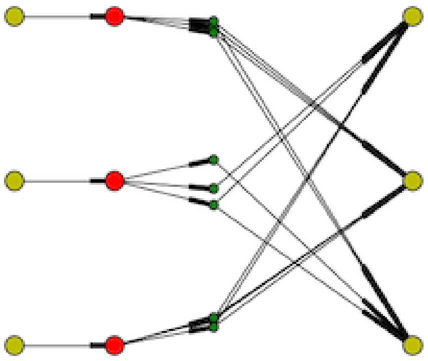

Figure 3: An example DAG Bitcoin Address Network, $\mathcal{A}_2$

This graph has no cycles, and there are fewer maximally connected subcomponents. It is a Directed Acyclic Graph (DAG). DAGs are ideal when optimizing for anonymity because constructing an identity graph from $\mathcal{A}_2$ using the method of using maximally connected components would yield a comparatively error-prone result in comparison to constructing $\mathcal{I}$. This fact is the basis for the method of bitcoin transaction anonymization discussed in this paper. In the following sections, we discuss graph characteristics preventing anonymization and address them individually by creating a hypothetical mixing service that takes each one into account.

2 Anonmynity Impediments

2.1 Maximal Connected Components

A maximal connected component of a directed graph is the largest connected subgraph of the larger graph [3]. A large number of connected subcomponents within an address network, $\mathcal{A}$, allows one to easily construct an identity graph, $\mathcal{I}$. Therefore, to optimize for anonymity, any mixing service designed to do so must be constructed with the goal of minimizing the number of connected components within its address set. The following items must be considered when minimizing a mixing service's maximal connected components.

Cycles & Cliques It is imperative that no cycles exist within the mixing service's address set. Cycles can easily become cliques which lead to a large number of connected subcomponents within $\mathcal{A}$. A clique of a directed graph is defined by a complete subgraph in which every pair of distinct nodes is connected by a pair of unique edges [2]. For analysis, a directed address graph would be converted to its corresponding undirected graph so that a clique could be formed without needing two edges between each node. In this way, the number of cliques in an address graph can serve as an upper bound for the number of nodes in its corresponding identity graph. In Figure 1, $\mathcal{A}$ had 8 cliques. Its identity graph, $\mathcal{I}$ had 6 nodes. Optimization for cliques has less to do with minimizing the number of cliques in $\mathcal{A}$, but rather, has more to do with minimizing the ratio of the number of cliques to the number of nodes. I.e.:

$$\frac{c}{|\mathcal{A}|}$$

where c is the number of cliques in $\mathcal{A}$ and $|\mathcal{A}|$ is the magnitude (number of nodes) of $\mathcal{A}$. We will refer to this this ratio as $\bar{c}$ later on.

Edges A large number of edges on any given node within an address set makes $\mathcal{A}$ extremely susceptible to large connected subcomponents. Therefore it is crucial that a mixing service minimizes the number of edges on any given address in its address set. For reasons discussed later, unlike cycles, a system cannot be constructed that perfectly minimizes the number of edges in its address set. A node within a perfectly optimized address set would have a total of 2 edges (one for the receiving its initial balance, and one for depleting its balance completely).

2.2 Entropy

Time Delay To associate an identity graph with an address set, one method that can be used is temporal analysis [4]. By associating groups of inputs with a similarly sized group of outputs, an identity graph can be constructed. Therefore, it imperative that any implementation incorporate a randomized time delay between transactions.

Input contamination To ensure anonymity, a mixing service must ensure that there is no path from a user to a user's destination. Given a sufficiently large number of users, a reasonable enough level of entropy can be achieved as to make this possible. However, to fully ensure anonymity, a good implementation would delay payout until there are enough idle coins in the address set from a different parent[1]. This also has the added benefit of creating a random time delay.

In systems that *do* allow for input contamination, $\bar{c}$ would be higher than comparable systems that do not allow for input contamination. For example, in the system below, each horizontal row of nodes corresponds to a transaction. In this implementation, input contamination is allowed, and $\bar{c} = 2.2$.

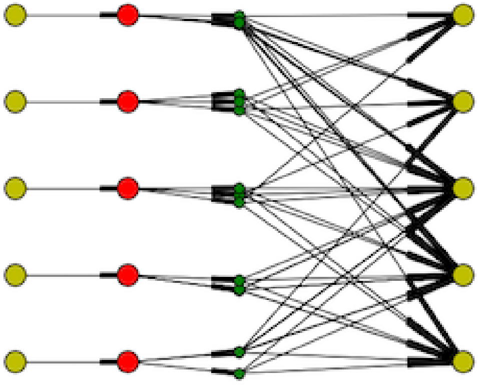

Figure 4: A poorly implemented mixer, where $\bar{c} = 2.2$

Now, contrast that with a system with the same number of nodes that does not allow for input contamination, and you can see that $\bar{c} = 1.63$.

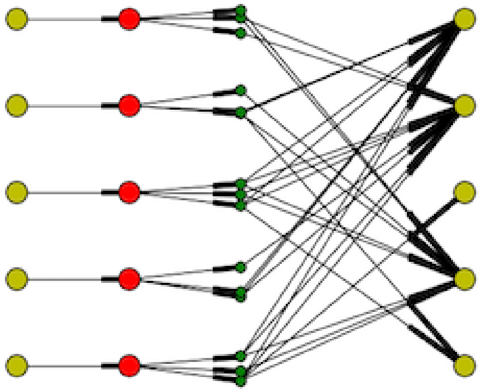

Figure 5: A comparable mixer, where input contamination is not allowed and $\bar{c} = 1.63$

3 Other Considerations

3.1 Trusted Third Party

Without completely overhauling the implementation of bitcoin itself, anonymization cannot be achieved without the use of a trusted third party[4]. That being said, there are certain things that can be done spread the reliance on a trusted third party over several entities.

3.2 Open Source

Any implementation should be open source. This ensures that the public is fully aware of the systems inner workings. This provides a means for peer-reviewal which in turn ensures that the system is not malicious. Any implementation that is not open source should be avoided like the plague.

3.3 Multiple APIs

The possession of wallets within the system should be distributed. The system should incorporate many APIs (Application Programming Interfaces) and also provide for a means to arbitrarily add and remove APIs in the event that a particular service becomes compromised. This way, no single person or organization is in possession of the wallets used in the system.

4 Security

Let $m(\alpha, \beta)$ be the function representing the mixing service where α is the origin address of the input transaction and β is the destination address of the mixed transaction. Let $\mathcal{A}$ be the existing address set already in the mixing service. The following two techniques can be used to compromise the mixing service.

4.1 $\alpha = \beta$ Attack

If $\alpha = \beta$ and the transaction is permitted by the mixing service, this could be the beginning of an introduction of a cycle into $\mathcal{A}$. Suppose $\mathcal{A}$ has had one previous transaction in which $\alpha \neq \beta$. Now suppose another transaction takes place where $\alpha = \beta$. A visualization is given in figure 4.

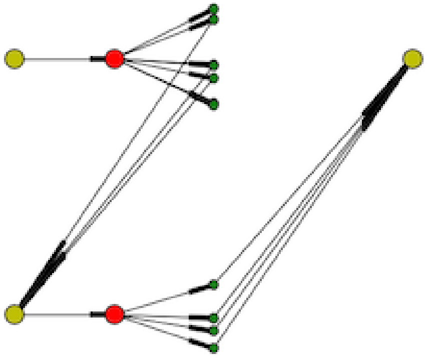

Figure 6: A transaction in which $\alpha = \beta$

Now, this on its own is not enough to introduce a cycle into $\mathcal{A}$, but if the attacker continued by targeting a $\beta \in \mathcal{A}$, a cycle would be produced.

4.2 $\beta \in \mathcal{A}$ Attack

Now, if the same attacker were to continue by using the mixing service to send bitcoin to an address which he knows lies within $\mathcal{A}$, a cycle would be produced. A visualization of this is given in figure 5.

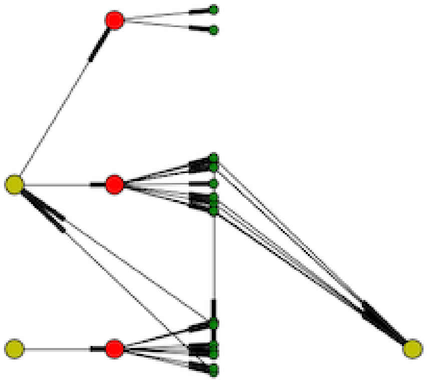

Figure 7: A transaction in which $\beta \in \mathcal{A}$ after a transaction in which $\alpha = \beta$

An attacker could continue doing this until he has contaminated a large portion of the address set with cycles.

4.3 Prevention

Any implementation should take careful consideration as to disallow transactions like the ones mentioned above. Repeated use of this technique could severely compromise the identity of the system's users, as well as its operators.

References

[1] Malte Möser. Anonymity of bitcoin transactions. In *Münster Bitcoin Conference*, 2013.

[2] University of California Davis. Clique.

[3] National Institute of Standards and Technology. Maximally connected component, September 2014.

[4] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. 07 2011.