Shared Send Untangling in Bitcoin

White Paper

Yuriy Yanovich[*] Pavel Mischenko[*] Aleksei Ostrovskiy[*]

Aug 21, 2016 (Version 1.0)

Abstract

Bitcoin is a widely used pseudo-anonymous cryptocurrency. Information about all transactions is stored in the public domain, whereas connections between transacting entities in the Bitcoin network and corresponding real-world entities are private. It is possible to reduce anonymity of the currency by analyzing history of operations and flow of funds using additional information (such as some facts that Bitcoin users have disclosed themselves). Therefore, additional measures to obfuscate transaction history (tangling) are popular among the network users. There are two basic approaches for such tangling: shared send and shared coin mixers.

In the paper, the concept of shared send transactions untangling is formalized. We prove NP-completeness of the shared send mixers detection problem and propose a number of practical modifications to the problem. Finally, we provide the results of our computational experiments to detect mixing schemes.

[*] crystal@bitfury.com, Bitfury Group

Bitcoin is a widely used cryptocurrency supported by the eponymous decentralized network [1]. The medium of exchange for Bitcoin is a payment unit with the same name. Bitcoins are transferred using transactions that utilize common cryptographic primitives (such as digital signatures and hash functions) to provide authentication capabilities. All transactions are included into a publicly available distributed ledger (the *blockchain*) after being verified by the nodes in the Bitcoin network.

One of important properties of Bitcoin is its pseudonymity [2]. Participants in the Bitcoin network are not obliged to disclose ownership of bitcoins; thus, data about the owners of bitcoins is generally not available. However, given the transaction history and data that Bitcoin users have disclosed about themselves, it may be possible to recover information about particular bitcoins. The uncovered information could be used for transaction risk scoring performed by Bitcoin financial services, during investigations by law enforcement, and in a variety of other applications.

In order to counteract loss of privacy associated with blockchain analysis attempts, the Bitcoin community has developed additional methods of ensuring pseudonymity. These methods are referred to as *mixing* or *transaction tangling*, as they operate by tangling, or mixing transaction histories from multiple sources. Thus, relationships among real-world entities owning addresses are obfuscated. Two main approaches can be distinguished among the existing tangling techniques:

- **Shared coin services.** Bitcoins are sent to an account of an intermediary service. After some time has passed, the service sends other available funds to the recipient's address. The process of sending to and reception from the intermediary is usually divided into several transactions. A client needs to trust the service; on the other hand, if operation was completed successfully, the connection between the sending and receiving addresses is almost impossible to trace. Shared coin services can be provided by exchanges or services without registration depending on the desired degree of anonymity.
- **Shared send.** Users organize into groups (via an intermediary) and tangle their coins in a single transaction (e.g., within the framework of the CoinJoin algorithm [3]). The small set of senders and recipients may be known, but it is not clear how the funds are distributed among them.

In this paper, we concentrate on the analysis of shared send mixers, which can be defined as two related problems:

- How to determine whether a certain bitcoin transaction is a result of a shared send operation
- How to uncover value flows within shared send transactions.

Detecting and analyzing shared send mixing is an important problem for the entities engaging in blockchain analysis and/or research on blockchain privacy issues. Since shared send mixing is more tractable than shared coin mixers from theoretical and computational standpoints, it could be a good starting point for blockchain analysis in general.

Previous research. There are a number of academic papers studying the anonymity properties of the Bitcoin network. F. Reid et al. [4] present general ideas about tracking bitcoin flows in the network

and wallets clustering. In [5], authors examine the strengths and weaknesses of the Bitcoin privacy model, noting that it is possible to link Bitcoin addresses using several heuristics. D. Ron et al. calculate in [6] some basic network statistics and note the activity to obfuscate bitcoin flows. In [7], authors perform successful addresses clustering together with integration of off-chain data to boost quality of the proposed models. The paper [8] describes a system implementing address clustering, presented together with examples of investigations. Finally, M. Möser et al. [9] study shared coin mixers and conclude that some behavioral patterns may be revealed.

Shared send mixers constitute a subset of the efforts to increase the privacy of Bitcoin users[1]. The first publicly known shared send algorithm is CoinJoin, proposed by Gregory Maxwell in [3]. Multistage tangling algorithm CoinSwap was proposed in [11]. The paper [12] proposes a multi-step protocol, which is a technical modification of CoinJoin. The paper [13] describes a multi-step algorithm with an additional random keys schedule; the proposed algorithm produces tangling with extra property that the service does not know where each of its clients sent money.

There are few results about analyzing shared send transactions, the closest to our work being CoinJoinSudoku [14]. The authors note that transactions created by CoinJoin could be untangled; however, we are not aware of the publicly available implementation of the algorithm described in the work or analysis of its application to the blockchain data.

Our contribution. We establish a theoretical approach to shared send transaction analysis, which formulates the transaction untangling problem in terms of the graph theory. We also describe several practically important modifications to the untangling problem. By reducing the untangling problem to a well-known partition problem, we rigorously prove the computational complexity of shared send analysis problem. Namely, we prove that shared send analysis is NP-complete.

We propose a following categorization of bitcoin transactions based on the outcome of the shared send analysis:

- **Simple transaction:** A transaction not produced as a result of a shared send operation (i.e., authored by a single entity). All other types of transactions are collectively referred to as *tangled* transactions
- **Separable transaction:** A tangled transaction allowing a unique separation into several sub-transactions representing money flows within the transaction
- **Ambiguous transaction:** A tangled transaction that admits at least two different interpretations as to the money flows
- **Intractable transaction:** A simple, separable or ambiguous transaction, the exact category of which cannot be discerned because of computational limitations (recall that shared send analysis is proven by us to be an NP-complete problem). Unlike other introduced categories,

[1] There are efforts in this direction unrelated to shared coin or shared send algorithms. For example, ZeroCash [10] is a scheme, in which privacy of users is protected by cryptographically obfuscating senders/recipients and amounts of transferred money.

the set of intractable transactions depends on the amount of computational resources at one's disposal; with unlimited resources, any intractable transaction would fall into one of the three previous categories.

Finally, we perform a large-scale computational experiment to categorize approximately 10 million bitcoin transactions. The results show that tangled transactions are a relatively frequent occurrence; further, about half of them may be untangled with moderate computational resources. Shared send analysis and other data mining tools form the core of the Bitfury's Crystal Blockchain service [15], which will be publicly available shortly after the release of the present white paper.

Structure of the paper. The paper is organized as follows:

- Section 1 gives the basic notation and assumptions, together with transaction examples. We also discuss practical modifications to the mixing problem.
- Section 2 contains the problem analysis from the theoretical perspective and the proof of its NP-completeness.
- Finally, we describe the setup and results of the computational experiments to detect shared send mixers in Section 3.

1 Definitions and Basic Assumptions

1.1 Bitcoin Transactions

As it was mentioned in the introduction, the Bitcoin Blockchain provides information about all bitcoins in use. This information, which presents the global state of the Bitcoin network, is a set of records (*unspent transaction outputs*, or *UTXOs* in the Bitcoin terminology). Each UTXO specifies the associated value in bitcoins and conditions under which these bitcoins may be spent.

Transfer of bitcoins is manifested through transactions. Each transaction contains:

- One or more inputs, each referring to a valid UTXO, and containing authentication information allowing to spend corresponding bitcoins
- One or more outputs, each containing a specification of a newly created UTXO.

After a completion of the transaction, UTXOs referenced by its inputs are removed from the UTXO set maintained by every Bitcoin node, and its outputs are added into the UTXO set. To incentivize inclusion of transactions into the Bitcoin Blockchain, the cumulative value of transaction outputs is usually slightly less than the cumulative value of its inputs. The difference – *transaction fee* – is paid to the Bitcoin node that included the transaction into the blockchain[2].

[2] This process is a part of the proof of work consensus algorithm established among nodes in the Bitcoin network; it is out of the scope of the present paper. See [1, 16] for more details.

In order to spend bitcoins, the owner generally needs to present to the network publicly verifiable authentication information proving that the assets belong to him. For convenience, conditions of spending a UTXO can be compressed with a collision-resistant hash function, thus forming a (Bitcoin) *address*. There are two main kinds of addresses used in Bitcoin today, corresponding to two kinds of conditions under which a UTXO may be spent:

- A knowledge of a single private key in the secp256k1 elliptic curve cryptosystem [17] may be required to spend a UTXO; in this case, the address is the hash of the corresponding public key.
- Otherwise, spending a UTXO may require providing proofs of knowledge of m out of n private keys ($1 \leq m \leq n$) in the same cryptosystem [18]; in this case, the address is a hash digest depending on m, n, and n public keys.

There may be multiple UTXOs associated with the same address. We will assume that:

- Each Bitcoin address is controlled by a single real-world entity. (Thus, we will ignore those sufficiently rare cases in which a multisignature address is used for joint ownership of bitcoins, and not for multi-factor authentication [19].)
- A single entity may control more than one address.

Definition 1. For the purpose of transaction analysis, a (Bitcoin) *transaction* is viewed an ordered triple $t = (\mathcal{A}, \mathcal{B}, c)$, consisting of:

- The finite multiset of transaction inputs $\mathcal{A}$, where each input $(a_i, A_i) \in \mathcal{A}$ is an ordered pair of the address A_i and the value of the input $a_i > 0$.
- The finite multiset of transaction outputs $\mathcal{B}$, where each output $(b_j, B_j) \in \mathcal{B}$ is an ordered pair of the address B_j and the value of the output $b_j \geq 0$.
- The transaction fee

$$c \equiv \sum_{(a_i,\cdot)\in\mathcal{A}} a_i - \sum_{(b_j,\cdot)\in\mathcal{B}} b_j \geq 0.$$

Definition 2. For an arbitrary multiset of transaction inputs or outputs $\mathcal{A}$:

- $\mathrm{Addr}(\mathcal{A})$ denotes the multiset of addresses in $\mathcal{A}$
- $\mathrm{Sum}(\mathcal{A})$ denotes the sum of values in $\mathcal{A}$,

that is,

$$\mathrm{Addr}(\mathcal{A}) = \{A : (\cdot, A) \in \mathcal{A}\}; \quad \mathrm{Sum}(\mathcal{A}) = \sum_{(a,\cdot)\in\mathcal{A}} a.$$

By construction, for any transaction $t = (\mathcal{A}, \mathcal{B}, c)$ it holds that $\mathrm{Sum}(\mathcal{A}) = \mathrm{Sum}(\mathcal{B}) + c$.

An address may be used several times in one transaction, as it may belong to the set of inputs and to the set of outputs. In addition, an address may occur several times in the set of inputs and/or outputs. For convenience, we simplify the multisets of inputs and outputs in all the following statements by performing the following transformation (Fig. 1):

1. Join all inputs belonging to the same address and all outputs belonging to the same address.
2. Put an address to the set of inputs if the value of the corresponding input is greater that the value of the output associated with the address (assuming the value is zero for addresses not encountered in inputs or outputs); otherwise regard that address as an output.
3. Recalculate the value for each new input and output straightforwardly.

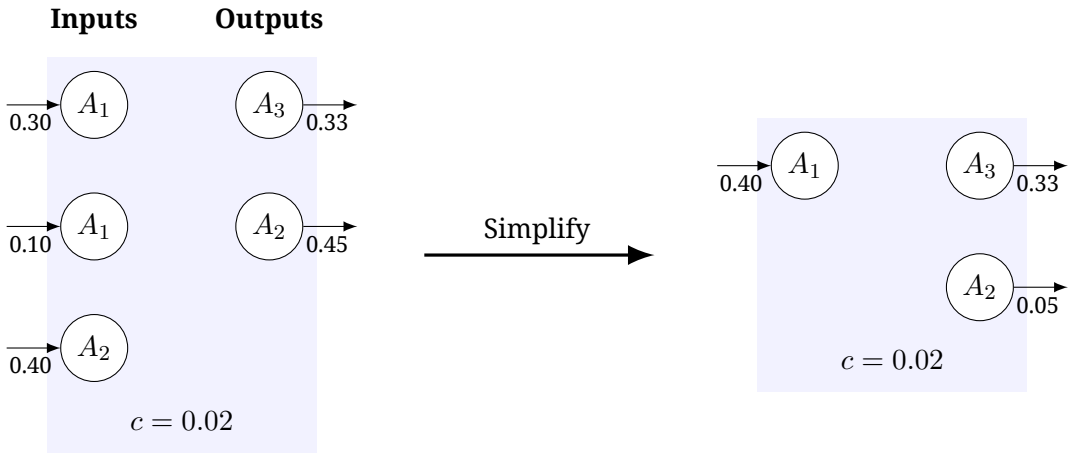

Figure 1: Simplification of a bitcoin transaction. Transaction inputs are represented by circled addresses (denoted by a capital letter and an index, e.g., A_3) with an inbound arrow and the label reflecting value associated with the input. Transaction outputs are similar to inputs, only with an outbound arrow. A transaction is represented with a rectangle encompassing one or more inputs, one or more outputs, and the fee c (assumed zero if not specified explicitly).

More formally, the described transformation is denoted $t' = \mathrm{Simplify}(t)$, where

$$t = (\mathcal{A}, \mathcal{B}, c); \quad t' = (\mathcal{A}', \mathcal{B}', c');$$
$$\mathcal{A}' = \{(A, -\mathsf{Bal}(t, A)) : \mathsf{Bal}(t, A) < 0\};$$
$$\mathcal{B}' = \{(A, \mathsf{Bal}(t, A)) : \mathsf{Bal}(t, A) > 0\};$$
$$c = c';$$

and the balance function Bal is defined on the address space as

$$\mathsf{Bal}(t, A) \equiv \sum_{(b_j, B_j) \in \mathcal{B}} b_j [B_j = A] - \sum_{(a_i, A_i) \in \mathcal{A}} a_i [A_i = A]$$

(the definition uses Iverson brackets [20]).

Remark 1. Any transformed transaction t' conforms to Definition 1 with the following additional constraints:

- Inputs $\mathcal{A}'$ and outputs $\mathcal{B}'$, as well as input addresses $\mathrm{Addr}(\mathcal{A}')$ and output addresses $\mathrm{Addr}(\mathcal{B}')$ are necessarily sets instead of generic multisets

- The sets of input and output addresses are disjoint: $\mathrm{Addr}(\mathcal{A}') \cap \mathrm{Addr}(\mathcal{B}') = \emptyset$.

For the sake of brevity, we will implicitly assume that all transactions in the following statement are transformed with the Simplify function. Correspondingly, we will generally omit any special notation for transformed transactions, and will sometimes refer to inputs/outputs by their address.

1.2 Shared Send Transactions

Definition 3. A transaction is called *shared send transaction* iff (if and only if) it has multiple inputs and multiple outputs after the previously described transformation Simplify: $|\mathcal{A}'| > 1$ and $|\mathcal{B}'| > 1$.

Note that a shared send transaction is not necessarily produced by a mixing service; the notion merely indicates a chance that a transaction was formed as a result of mixing. For example, non-mixing shared send transactions may occur as a result of the use of a hierarchical deterministic (HD) wallet [21]; HD wallets create a new address for each operation.

The idea of shared send mixers was formulated and implemented in the CoinJoin algorithm [3]. The algorithm can be informally summarized as "when you want to make a payment, find someone who also wants to make a payment and make a joint payment together." As such, a (CoinJoin) mixing transaction necessarily confirms to Definition 3; it can be viewed as a joint spending of bitcoins controlled by more than one entity.

One of the main objectives of transaction analysis, which we concentrate on in this paper, is to determine relationships among addresses. Namely, given two Bitcoin addresses A_1 and A_2, the goal is to determine based on the blockchain data whether these addresses belong to the same real-world entity, e.g., a natural person. One of the basic tools used for discovering address relations is the *common spending* heuristic [8]:

Assumption 1 (common spending)**.** All addresses referenced in the transaction inputs belong to the same entity.

This assumption is generally sound for non-mixing transactions, because such transactions are commonly created and authored by a single entity. In contrast, for shared send mixing transactions, transaction inputs belong to several entities. Applying the common spending heuristic to mixing transactions thus may lead to incorrect conclusions (e.g., wrongfully attributing addresses belonging to different entities, to the same entity).

A mixing transaction $t = (\mathcal{A}, \mathcal{B}, c)$ can be represented by a bipartite graph $G(t)$, which reflects value flows among participants (Fig. 2, 3). This transaction graph is constructed as follows:

- Vertices correspond to transaction inputs and outputs; one part consists of inputs and another part consists of outputs. All vertices have positive weights that correspond to amounts of bitcoins associated with a corresponding transaction input or output.
- Each edge connects a certain input vertex with a certain output vertex. Two vertices are connected by an edge if the entity controlling the input authored spending from the input to the output; the output may belong to the same entity or to a different entity.

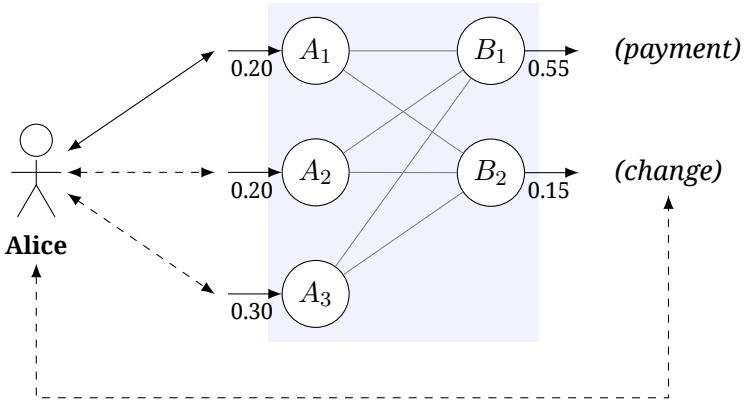

Figure 2: Structure of a typical non-mixing transaction in Bitcoin. All value on the transaction inputs are controlled by the same entity, Alice. Alice authenticates spending from the inputs to a payment address B_1 belonging to a third party (e.g., a retailer) and to the *change address* B_2 controlled by Alice. If an external observer knows that A_1 is controlled by Alice (e.g., because Alice has posted this address on a public forum), he may infer that A_2 and A_3 also are controlled by Alice. Furthermore, it may be conjectured that B_2 (and not B_1) is the change address based on the output values.

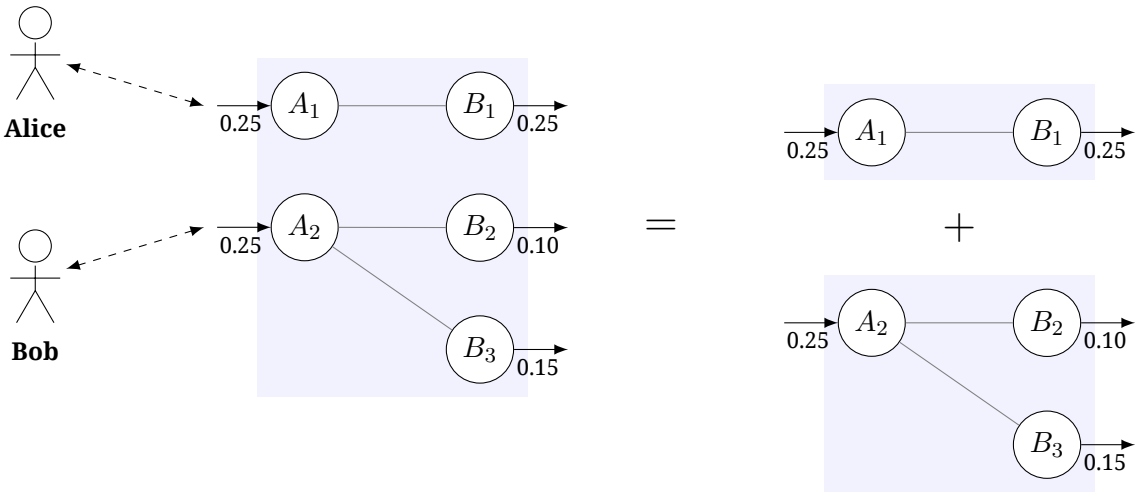

Figure 3: Shared send mixing transaction with two participants, Alice and Bob, represented as a bipartite graph. Alice spends 0.25 bitcoins to the address B_1. Bob spends 0.25 bitcoins to the addresses B_2 and B_3. The transaction could be viewed as a composition of two sub-transactions authored by Alice and Bob respectively.

More formally,

$$G(t) = (V, E); \quad V = \mathcal{A} \cup \mathcal{B}; \quad E \subseteq \mathcal{A} \times \mathcal{B}.$$

Each transaction input node is connected to at least one output node:

$$\forall (a_i, A_i) \in \mathcal{A}\ \exists (b_j, B_j) \in \mathcal{B} : ((a_i, A_i), (b_j, B_j)) \in E.$$

For non-mixing transactions (Fig. 2), the transaction graph is complete, i.e., each input is connected to each output of the transaction. This corresponds to the fact that all inputs of the transaction are

controlled by the same entity, who authored value flow to all output addresses of the transaction (which may or may not belong to this entity). On the other hand, if the transaction is a shared spend mixing transaction (Fig. 3), its graph is incomplete, as there are multiple entities controlling transaction inputs. Each connected component in the graph $G(t)$ is a complete bipartite graph that corresponds to a single entity participating in the mixing transaction.

Public information about the transaction contains all data about the vertices of the transaction graph, but does not contain any information about the edges. Thus, the goal of shared send analysis is to reconstruct the edges based on the vertices and, perhaps, some other information. Recovering information about the edges is an important subtask of the previously stated problem of finding relationships among addresses; therefore, it would be useful in a variety of applications, such as transaction risk scoring and law enforcement investigations.

Problem 1 (shared send analysis)**.** Based on the transaction $t = (\mathcal{A}, \mathcal{B}, c)$, reconstruct edges of the transaction graph $G(t)$.

Problem 2 (mixer detection)**.** Based on the transaction $t = (\mathcal{A}, \mathcal{B}, c)$, determine whether edges of $G(t)$ can be reconstructed uniquely.

As a means to simplify analysis without losing its accuracy for most real-world cases, we assume that participants in a mixing transaction do not have mutual financial arrangements beyond the transaction itself.

Assumption 2. Intended expenses of each input subset in the transaction do not exceed their actual expenses. In other words, there are no financial arrangements beyond the transaction, such as "Party A credits Party B with a certain amount of bitcoins in the mixing transaction, which is to be returned by Party B."

Assumption 2 imposes restrictions on edge placement in a transaction graph. Namely, the graph must satisfy the following restriction on connected components.

Assumption 3. For any connected component in the transaction graph $G(t)$, if $\mathcal{A}$ is the set of inputs in the component, and $\mathcal{B}$ is its set of outputs, then $\mathrm{Sum}(\mathcal{A}) \geq \mathrm{Sum}(\mathcal{B})$.

Based on Assumption 3, it is possible to describe possible connected components in the transaction graph.

Definition 4. Consider a transaction $t = (\mathcal{A}, \mathcal{B}, c)$. A pair of non-empty sets $(\mathcal{A}', \mathcal{B}')$, where $\mathcal{A}' \subset \mathcal{A}$, $\mathcal{B}' \subset \mathcal{B}$, is called *connectable* in t iff the following condition holds:

$$\mathrm{Sum}(\mathcal{B}') + c \geq \mathrm{Sum}(\mathcal{A}') \geq \mathrm{Sum}(\mathcal{B}').$$

That is, connectivity means a reasonable assumption that bitcoins from the addresses in $\mathcal{A}'$ were spent to the addresses in $\mathcal{B}'$, and from $\mathcal{A} \setminus \mathcal{A}'$ to $\mathcal{B} \setminus \mathcal{B}'$.

Definition 5. A collection of sets $\{X_k\}_{k=1}^K$ is called a *partition* of a set X if the sets in the collection are pairwise disjoint, and their union equals X. We denote a partition as $X = \bigsqcup_{k=1}^K X_k$.

Definition 6. Consider a transaction $t = (\mathcal{A}, \mathcal{B}, c)$. A pair of K-element partitions ($K > 1$) of input and output sets, i.e. $\mathcal{A} = \bigsqcup_{k=1}^K \mathcal{A}_k$, $\mathcal{B} = \bigsqcup_{k=1}^K \mathcal{B}_k$ is called *acceptable* iff each pair $(\mathcal{A}_k, \mathcal{B}_k)$ is connectable.

We will often denote partitions in the form $\mathcal{P} = \{(\mathcal{A}_1, \mathcal{B}_1), \ldots, (\mathcal{A}_K, \mathcal{B}_K)\}$, and say that sets $\mathcal{A}_i$ and $\mathcal{B}_i$ in the partition *correspond* to each other. See Figure 4 for the example of acceptable and unacceptable partitions.

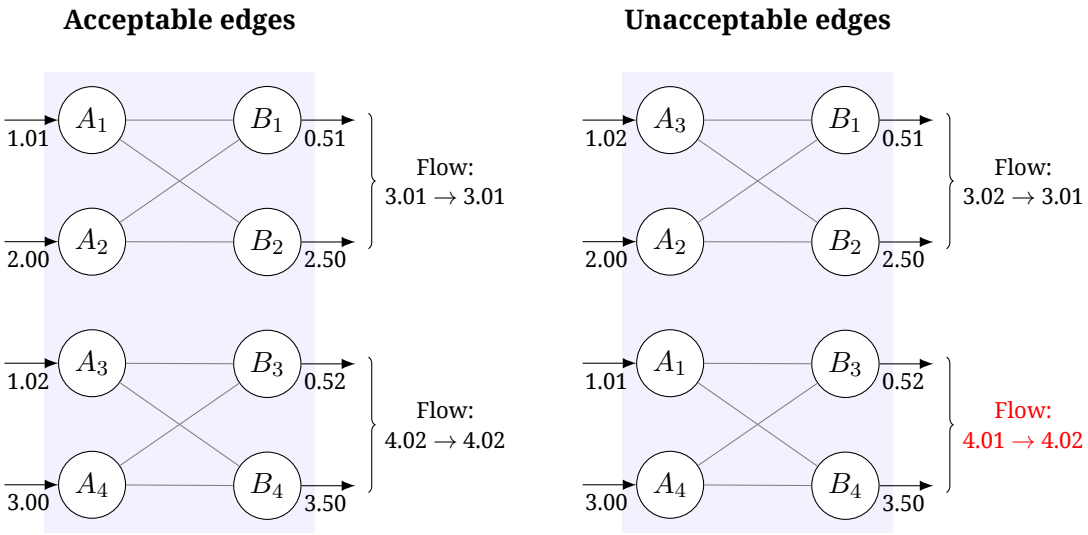

Figure 4: Two edge placements for a transaction graph with the same vertices. The left one is acceptable (the sum of upper input amounts equals to the sum of upper output amounts, and the same is true for the lower sums). The right one is not acceptable, because the sum of bottom inputs A_1 and A_4 is less than the sum of bottom outputs B_3 and B_4.

Intuitively, an acceptable partition represents a possible way of transaction untangling. Indeed, from Assumption 2 one can infer that bitcoins from the set of addresses in $\mathcal{A}_k$ were sent to the addresses in $\mathcal{B}_k$, $k = 1, \ldots, K$. Thus, Problem 1 may be reformulated as follows.

Problem 3 (untangling)**.** Based on the transaction $t = (\mathcal{A}, \mathcal{B}, c)$, produce all acceptable partitions of the transaction graph $G(t)$.

In order to reduce complexity of Problem 3, we need to restrict the search space. We do this by introducing the concept of a minimal partition.

Definition 7. A connectable pair $(\mathcal{A}, \mathcal{B})$ is called *minimal* if the sets in it cannot be represented as $\mathcal{A} = \mathcal{A}_1 \sqcup \mathcal{A}_2$, $\mathcal{B} = \mathcal{B}_1 \sqcup \mathcal{B}_2$ such that pairs $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_2, \mathcal{B}_2)$ are connectable.

Definition 8. An acceptable partition $\mathcal{P} = \{(\mathcal{A}_k, \mathcal{B}_k)\}_{k=1}^K$ is called *minimal* if it cannot be further subdivided into an acceptable partition, i.e., $(\mathcal{A}_k, \mathcal{B}_k)$ is a minimal connectable pair for each $k = 1, \ldots, K$.

Remark 2. Consider an acceptable partition. If it is not minimal, then it contains a non-minimal pair $(\mathcal{A}, \mathcal{B})$. This pair can be decomposed as $\mathcal{A} = \mathcal{A}_1 \sqcup \mathcal{A}_2$, $\mathcal{B} = \mathcal{B}_1 \sqcup \mathcal{B}_2$ such that pairs $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_2, \mathcal{B}_2)$ are connectable. We can remove $(\mathcal{A}, \mathcal{B})$ from the partition and put $(\mathcal{A}_1, \mathcal{B}_1), (\mathcal{A}_2, \mathcal{B}_2)$ into it, obtaining another acceptable partition (Fig. 5). We call this process the *extension* of the partition. Every acceptable partition can be extended to a minimal partition; indeed, the number of pairs in partition grows on each step of the extension process, and this number has an upper bound (obviously, the number of connectable pairs in any partition cannot exceed the number of transaction inputs or outputs). Note that a single partition may be extended into multiple minimal partitions.

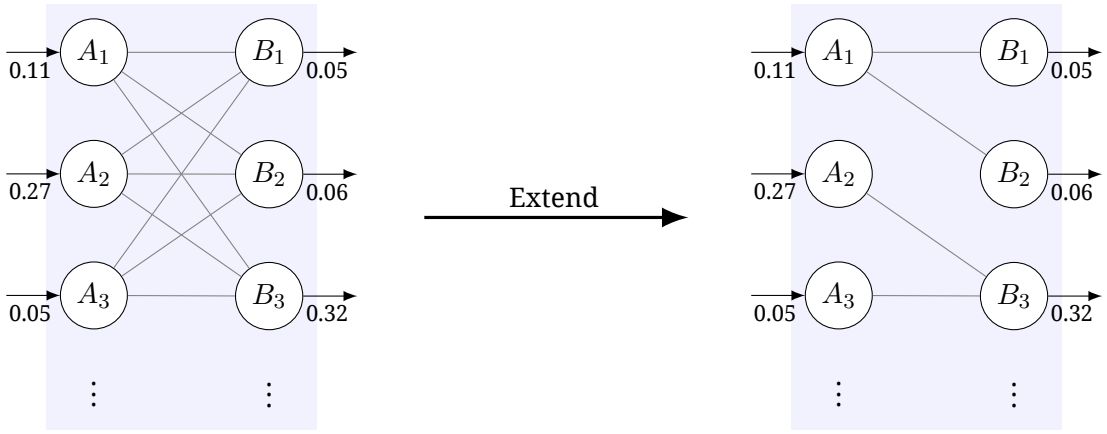

Figure 5: Non-minimal partition and one of the ways to extend it to a minimal one. Note that the extension in this case is not unique; it is possible to create a different minimal partition by connecting A_3 with B_1 and $\{A_1, A_2\}$ with $\{B_2, B_3\}$.

Using the concept of minimal partitions, Problem 3 is restricted as follows.

Problem 4 (minimal untangling)**.** Based on the transaction $t = (\mathcal{A}, \mathcal{B}, c)$, produce every minimal partition of the transaction graph $G(t)$.

By accepting such a restriction, we implicitly assume that a non-mixing transaction is unlikely to look like a mixing transaction. Furthermore, the value flows are more likely to correspond to a minimal acceptable partition than to a non-minimal partition extensible to this minimal partition, as additional measures need to be taken to make value flows dividable. In the case these assumptions do not hold (which would be a relatively rare occurrence), additional relationships among entities discovered after solving Problem 4 could be established through methods that are out of the scope of this paper.

Having finally established the problem in a strict and non-redundant way, we can now define different categories of shared send transactions.

Definition 9. Depending on the result of the minimal untangling formulation of the shared send analysis problem (Problem 4), a shared send transaction is called:

- *simple* iff it does not have a minimal acceptable partition (Fig. 2). Other shared send transactions will be jointly referred to as *tangled* or *mixing* shared send transactions
- *separable* iff a unique minimal partition exists (Fig. 6)
- *ambiguous* iff there are at least two different minimal partitions (Fig. 7).

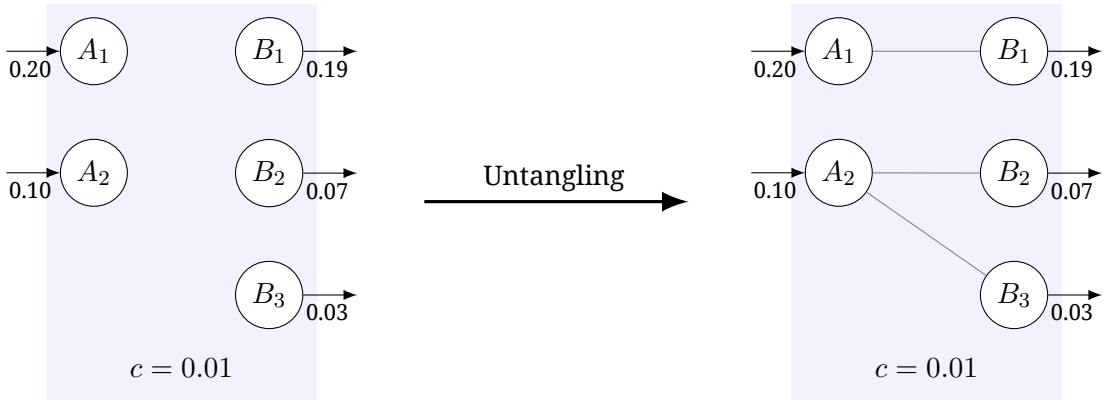

Figure 6: Separable transaction. As a result of transaction analysis, it is possible to conclusively separate the transaction into two sub-transactions, associating address A_1 with B_1, and A_2 – with B_2 and B_3.

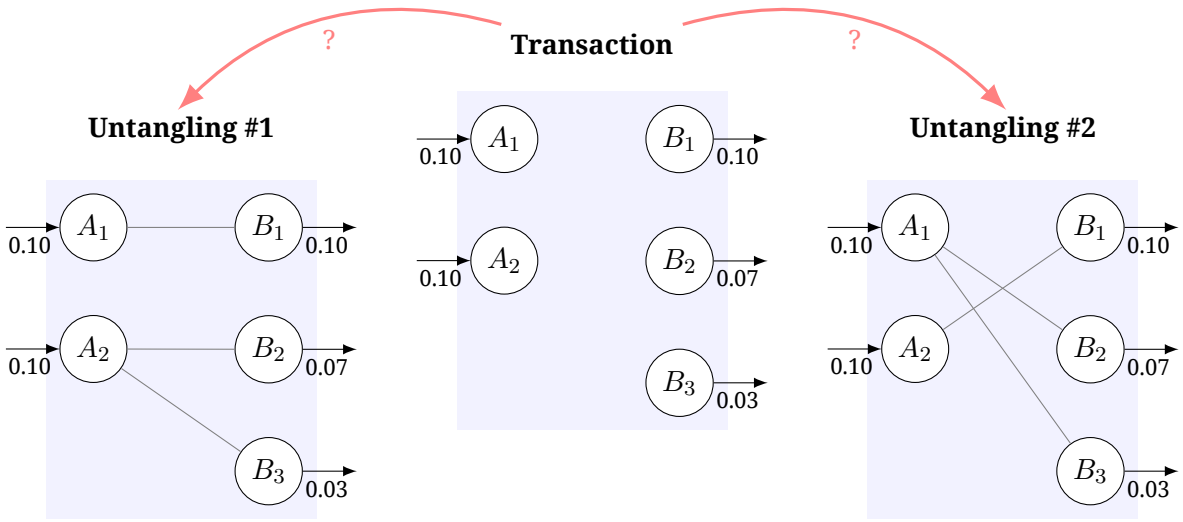

Figure 7: Ambiguous transaction that admits two different untangling outcomes (i.e., two minimal partitions of the transaction graph). It is impossible to determine without any additional data, whether the owner of address A_1 spent his funds to the address B_1 or to the addresses B_2 and B_3.

Additionally, we semi-formally define *intractable transactions* as tangled transactions, for which attribution to a specific category is infeasible because of computational limitations inherent to any practical solver of Problem 4. Unlike with categories described in Definition 9, which transactions categorize as intractable depends on capabilities of a concrete solver. For a solver with infinite computational resources, all transactions would be tractable.

Note that Definition 9 can be trivially generalized to encompass all bitcoin transactions (Fig. 8). Namely, all non-shared send transactions fall into the category of simple transactions.

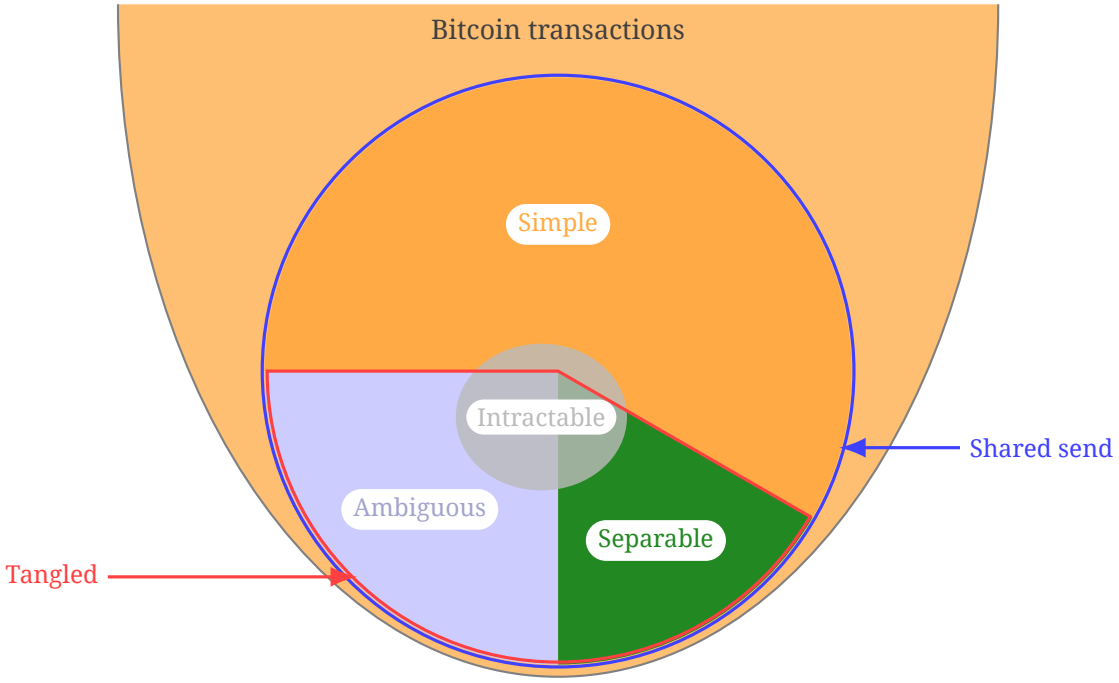

Figure 8: Categories of bitcoin transactions with relation to the untangling problem. Basic categories (simple, separable, ambiguous and intractable transactions) are marked with fill; aggregate categories (shared send and tangled transactions) – with a colored contour.

1.3 Modifications

In this section we list possible modifications to Assumption 2 and Definition 9 and situations where these modifications could be applicable.

1.3.1 Grouping Addresses

A situation where a person or another entity owns several addresses is widespread in the Bitcoin network. If this information is known (e.g., pre-calculated), it is natural to merge inputs and outputs belonging to the same owner, like we did with addresses over the course of the Simplify transformation (Fig. 9). As all acceptable partitions for a transaction transformed in this way are acceptable for the original transaction, the transformation can only clarify value flows; it cannot transform a separable or simple transaction into an ambiguous one.

1.3.2 Discarding Small Inputs and Outputs

If we want to understand the flow of bitcoins, it is sometimes reasonable to leave small amounts out of consideration. In our problem this leads to the following idea: Consider a transaction which contains some inputs that are smaller than fee. Then we may exclude them from inputs set while preserving the structure of the transaction graph. After this transformation, an ambiguous transaction may become separable (Fig. 10). Similar procedure applies for outputs, though it is more technical as described below.

Under Assumption 2, the transaction is ambiguous if the following conditions hold:

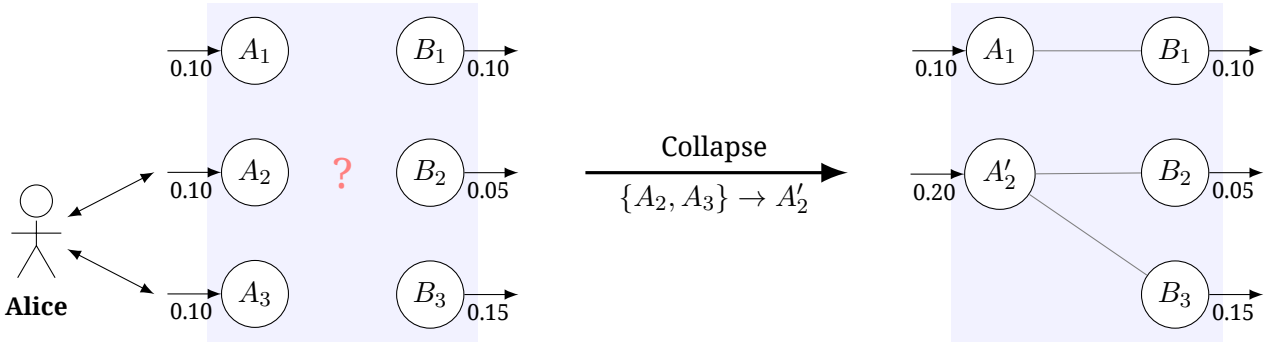

Figure 9: Transforming a transaction using information about address grouping. A previously ambiguous transaction becomes separable because it is known that two of input addresses, A_2 and A_3, belong to the same entity, Alice, and therefore may be collapsed into a single pseudo-address.

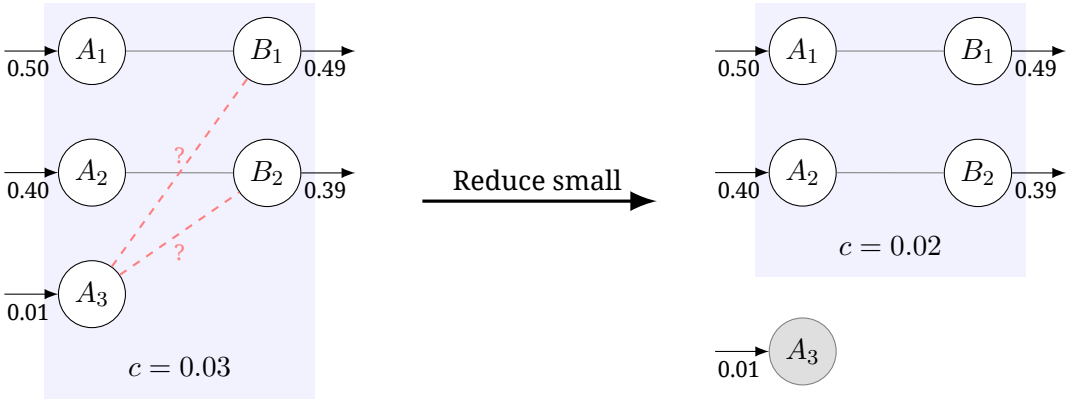

Figure 10: Transforming a transaction graph by removing small inputs and outputs. The source transaction is ambiguous (indeed, it is possible to ascribe the input A_3 to the either output). After the removal of the small input, the transaction becomes separable.

- the transaction contains an input which is smaller than or equal to the fee (the "small" input)
- at least one acceptable transaction partition exists.

Indeed, the "small" input can be added to any input subset in the partition, so that at least two different minimal partitions will be formed. Similarly, a transaction is ambiguous if the following two conditions hold:

- the transaction contains a "small" output of value ε
- there exists at least one acceptable partition of inputs and outputs (excluding the "small" output) $\{(\mathcal{A}_k, \mathcal{B}_k)\}_{k=1}^{K}$, such that for each k $\mathrm{Sum}(\mathcal{A}_k) \geq \mathrm{Sum}(\mathcal{B}_k) + \varepsilon$.

Such transactions can be excluded from the category of ambiguous transactions by iteratively removing small inputs and then small outputs from the transaction and adjusting the transaction fee correspondingly (Algorithm 1). It is possible to change the order in pre-processing procedure (i.e., by considering transaction outputs first), which may yield slightly different results in untangling.

Algorithm 1: Discarding small inputs and outputs

Input: Transaction $t = (\mathcal{A}, \mathcal{B}, c)$

Output: Modified transaction t'

1: Initialize $t' := t$.
2: Arrange all transaction inputs in t' in the ascending order of their value.
3: **for all** inputs $(a, A) \in \mathcal{A}$ **do**
4: **if** $a \leq c$ **then**
5: Remove (a, A) from the inputs of t' and modify its fee: $c := c - a$.
6: **else**
7: **break**
8: Perform ordinary processing for the transaction t' with a modified fee and input set to determine all its minimal acceptable partitions $\{\mathcal{P}_i\}$.
9: Calculate

$$\delta = \max_{\mathcal{P}_i} \min_{k=1,\ldots,K_i} (\mathrm{Sum}(\mathcal{A}_{i,k}) - \mathrm{Sum}(\mathcal{B}_{i,k})),$$

where $(\mathcal{A}_{i,k}, \mathcal{B}_{i,k})$ is the k-th minimal connectable pair in the partition $\mathcal{P}_i$.
10: Remove all outputs with the value less than δ from the output set of t' and add their total value to the value of the fee.
11: **return** the obtained transaction t'.

1.3.3 Approximate Equalities

If a shared send transaction is performed by a mixing service, the service may take a certain fee. This generally results in a simple transaction as per Definition 9. To take such cases into account, we explicitly introduce parameter $r \in [0, 1]$ – a relative fee rate charged by the service (Fig. 11). The formulation of Problem 4 remains the same, with the following exception: a pair of sets $(\mathcal{A}', \mathcal{B}')$ is now called connectable iff

$$\frac{1}{1-r}\,\mathrm{Sum}(\mathcal{B}') + c \geq \mathrm{Sum}(\mathcal{A}') \geq \mathrm{Sum}(\mathcal{B}').$$

Note that the changed formulation may lead to a minimal partition in which the transaction output(s) associated with mixer fees are not connected to the transaction inputs.

2 NP-Completeness of Untangling Problem

Given a concrete mathematical formulation of the mixer detection problem (Problem 2), we can obtain an expected (though unpleasant from a computational point of view) result that this problem is NP-complete. First, we prove two lemmas which give us sufficient conditions for a transaction to be ambiguous. Then we prove that they are necessary in a sense that for an ambiguous transaction at least one of these conditions holds. Because the introduced conditions support polynomial verification, this provides the proof that the considered problem lies in NP.

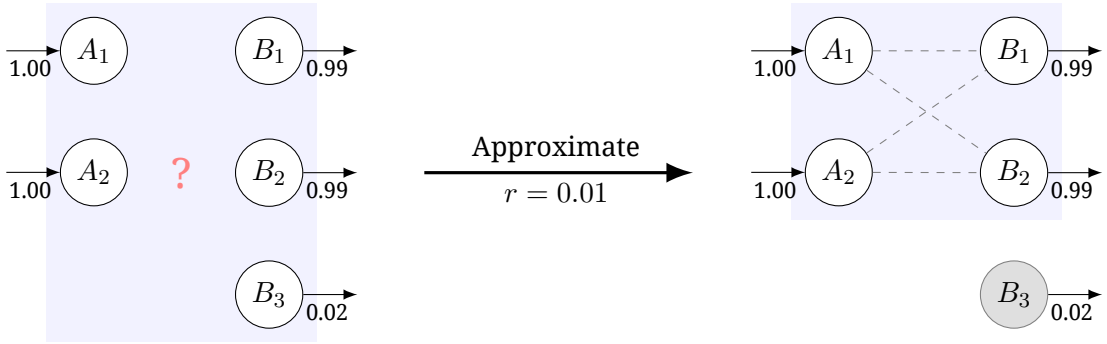

Figure 11: Transformation of a transaction that accounts for a fee taken by the mixing service, which corresponds to the output B_3. Assuming the mixing fee rate $r = 0.01$, the transaction becomes ambiguous, as it logically should.

Lemma 1. *Consider a transaction $t = (\mathcal{A}, \mathcal{B}, c)$. Suppose that at least one of the following conditions holds:*

1. *There are two different subsets of inputs $\mathcal{A}_1, \mathcal{A}_2 \subset \mathcal{A}$ and a subset of outputs $\mathcal{B}_1 \subset \mathcal{B}$, such that $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_2, \mathcal{B}_1)$ are connectable.*
2. *There are two different subsets of outputs $\mathcal{B}_1, \mathcal{B}_2 \subset \mathcal{B}$ and a subset of inputs $\mathcal{A}_1 \subset \mathcal{A}$, such that $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_1, \mathcal{B}_2)$ are connectable.*

Then the transaction is ambiguous under Definition 9.

Proof.

Without loss of generality, suppose that the first condition is true. From the connectivity property, $\{(\mathcal{A}_k, \mathcal{B}_1), (\mathcal{A} \backslash \mathcal{A}_k, \mathcal{B} \backslash \mathcal{B}_1)\}$ are acceptable partitions for $k = 1, 2$. As was discussed in Remark 2, each of them can be extended to a minimal partition. If the obtained minimal partitions are equal, then $\mathcal{B}_1$ can be uniquely presented as union of output sets in this partition $\mathcal{P}$: $\mathcal{B}_1 = \mathcal{B}_1^{min} \sqcup \mathcal{B}_2^{min} \sqcup \cdots \sqcup \mathcal{B}_L^{min}$, $L \geq 1$. In this case, any input set corresponding to $\mathcal{B}_1$ is uniquely determined as a union of the corresponding input sets in $\mathcal{P}$:

$$\forall k \in \{1, 2\} \; \mathcal{A}_k = \bigsqcup_{l=1}^{L} \mathcal{A}_l^{min}, \quad (\mathcal{A}_l^{min}, \mathcal{B}_l^{min}) \in \mathcal{P},$$

which contradicts the fact that $\mathcal{A}_1$ and $\mathcal{A}_2$ are different. Thus, t allows two minimal acceptable partitions, i.e., this transaction is ambiguous. ∎

Example. The transaction depicted in Fig. 7 is ambiguous. Indeed, the conditions of Lemma 1 hold for $\mathcal{A}_1 = \{A_1\}, \mathcal{A}_2 = \{A_2\}, \mathcal{B}_1 = \{B_1\}$.

Lemma 2. *Consider a transaction $t = (\mathcal{A}, \mathcal{B}, c)$. Suppose at least one of the following conditions holds:*

1. *There exist two pairs of subsets $\mathcal{A}_1, \mathcal{A}_2 \subset \mathcal{A}$ and $\mathcal{B}_1, \mathcal{B}_2 \subset \mathcal{B}$, such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are different with a non-empty intersection, and both $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_2, \mathcal{B}_2)$ are minimal connectable pairs as per Definition 7.*

2. *There exist two pairs of subsets $\mathcal{A}_1, \mathcal{A}_2 \subset \mathcal{A}$ and $\mathcal{B}_1, \mathcal{B}_2 \subset \mathcal{B}$, such that $\mathcal{B}_1$ and $\mathcal{B}_2$ are different with a non-empty intersection, and both $(\mathcal{A}_1, \mathcal{B}_1)$ and $(\mathcal{A}_2, \mathcal{B}_2)$ are minimal connectable pairs.*

Then the transaction is ambiguous under Definition 9. (Note that $\mathcal{B}_1$ and $\mathcal{B}_2$ in condition 1 and $\mathcal{A}_1$ and $\mathcal{A}_2$ in 2 are not necessarily different.)

Proof.

Without loss of generality, suppose that the first condition is true. Consider minimal partitions obtained from $\{(\mathcal{A}_k, \mathcal{B}_k), (\mathcal{A} \setminus \mathcal{A}_k, \mathcal{B} \setminus \mathcal{B}_k)\}$ for $k = 1, 2$ by the process of extension (see Remark 2). They are different, because the pair $(\mathcal{A}_1, \mathcal{B}_1)$ is contained in the first minimal partition, but it cannot be present in the second minimal partition. Indeed, the second minimal partition contains the pair $(\mathcal{A}_2, \mathcal{B}_2)$, and since $\mathcal{A}_1$ has at least one vertex in common with $\mathcal{A}_2$, it is impossible for this minimal partition to contain $\mathcal{A}_1$ as well. ■

Example. Consider a transaction depicted in Fig. 4. It is easy to verify that every subset of the inputs and every subset of the outputs yields a different sum. Thus, the conditions of Lemma 1 do not hold. However, conditions of Lemma 2 hold for $\mathcal{A}_1 = \{A_1, A_2\}$, $\mathcal{B}_1 = \{B_1, B_2\}$, $\mathcal{A}_2 = \{A_1, A_4\}$ and $\mathcal{B}_2 = \{B_1, B_4\}$, therefore the transaction is ambiguous. Indeed,

$$\big\{(\{A_1, A_2\}, \{B_1, B_2\}), (\{A_3, A_4\}, \{B_3, B_4\})\big\}$$

and

$$\big\{(\{A_1, A_4\}, \{B_1, B_4\}), (\{A_2, A_3\}, \{B_2, B_3\})\big\}$$

are different minimal partitions of the transaction.

Theorem 1. *If a transaction $t = (\mathcal{A}, \mathcal{B}, c)$ is ambiguous as per Definition 9, then at least one of the conditions listed in Lemmas 1 and 2 holds.*

Proof.

As the transaction is ambiguous, there exist are two minimal partitions

$$\mathcal{P} = \{(\mathcal{A}_1, \mathcal{B}_1), ..., (\mathcal{A}_K, \mathcal{B}_K)\}; \quad \mathcal{P}' = \{(\mathcal{A}'_1, \mathcal{B}'_1), ..., (\mathcal{A}'_{K'}, \mathcal{B}'_{K'})\}.$$

Suppose there is a set $\mathcal{A}_i$ from the partition $\mathcal{P}$ which does not belong to $\mathcal{P}'$. In this case, pick an element $a \in \mathcal{A}_i$ and find a set $\mathcal{A}'_j$ which contains a. It is clear that the first condition of Lemma 2 holds for $(\mathcal{A}_i, \mathcal{B}_i), (\mathcal{A}'_j, \mathcal{B}'_j)$.

Suppose that each $\mathcal{A}_i$ is present in the second partition; without a loss of generality, $\mathcal{A}_i = \mathcal{A}'_i$. As partitions are different, there is a set $\mathcal{A}_j$ such that its corresponding output set $\mathcal{B}'_j$ in $\mathcal{P}'$ is different from the set $\mathcal{B}_j$ in $\mathcal{P}$. Correspondingly, conditions of Lemma 1 hold for $\mathcal{A}_j$, $\mathcal{B}_j$, $\mathcal{B}'_j$. ■

Now we are ready to prove the central theoretical result of our work.

Theorem 2. *The problem of detection of ambiguous shared send transactions (i.e., Problem 2) is NP-complete under Definition 9 and Assumption 2.*

Proof.

First, Problem 2 lies in the class NP. In order to verify a solution to Problem 2 for a transaction $t = (\mathcal{A}, \mathcal{B}, c)$, it is sufficient to present three subsets satisfying the conditions of Lemma 1, or to present four subsets satisfying the conditions of Lemma 2. Any of these checks is feasible in $\mathcal{O}(|\mathcal{A}| + |\mathcal{B}|)$ time and requires $\mathcal{O}(|\mathcal{A}| + |\mathcal{B}|)$ space, where $\mathcal{O}(\cdot)$ is big O asymptotic notation [22].

Second, observe that the (half) partition problem is reduced to the problem of detecting ambiguous transactions. The partition problem is NP-complete [23] and is formulated as follows: given a set of positive integers $s_1, \ldots, s_P$, find out whether there is a subset of indexes $I \subset \{1, \ldots, P\}$ such that

$$\sum_{p \in I} s_p = \frac{S}{2}; \quad S \equiv \sum_{p=1}^{P} s_p.$$

Let us polynomially reduce the partitioning problem to ambiguous transactions detection. Given data from the partitioning problem, construct a transaction $t = (\mathcal{A}, \mathcal{B}, c)$ serving as the input for Problem 2 as follows:

- Create a transaction input for each integer s_i: $\mathcal{A} = \{(s_i, A_i)\}_{i=1}^{P}$, where addresses A_i could be chosen arbitrarily
- Create two transaction outputs with equal value: $\mathcal{B} = \{(S/2, B_1), (S/2, B_2)\}$, with B_1 and B_2 chosen arbitrarily
- Set transaction fee $c = 0$.

This reduction is carried out in polynomial time. Furthermore, if the transaction is found to be ambiguous, there is a subset of the inputs such that their sum equals to $S/2$. Hence, these subsets could be returned as a solution to the partition problem in polynomial time. NP-completeness is proved. ∎

3 Experiments

We have analyzed approximately 10,000,000 bitcoin transactions taking place between May 27 and July 11, 2016, preprocessed as described in Section 1.3.2. Namely, we removed small values from the set of inputs as per steps 1–7 of Algorithm 1. Because of performance reasons, we did not remove transaction outputs as specified by steps 8–10 of the algorithm, as it would require to perform full shared send analysis for an intermediate transaction. We did not perform other heuristics described in Section 1.3; i.e., we did not group addresses in transactions based on off-chain data or preliminary analysis, and did not treat mixer service fees specially.

The algorithm solving Problem 4 was implemented in Python and C++; the SciPy [24] library was used to analyze the results. The calculations were performed on a dedicated server with a 12-core CPU and 64 GB available RAM.

The results of shared send analysis are presented in Fig. 12. Shared send transactions (Definition 3) constitute approximately 13.5% of all studied bitcoin transactions. Of these, about 82% are simple

transactions, ≈9.6% are separable, and ≈6.3% are ambiguous. Finally, the remaining 2.5% of shared send transactions are intractable given the computational constraints. The frequency ratios of various categories of transactions remain stable throughout the observed period.

Figures 13–15 display the distributions of the number of transaction inputs, outputs and the sum of these two values (i.e., the number of vertices in the transaction graph). Predictably, the more inputs and outputs, the more likely for a transaction to be ambiguous or intractable. The distributions in question manifest considerable positive skew (Table 1), i.e., they have a heavy left tail. The distributions for simple transactions are most skewed, while the distributions for intractable transactions are consistently least skewed of all. Note that there is a considerable asymmetry between the number of inputs and outputs in transactions: about a half of transactions in all categories have two outputs, while the number of inputs is distributed more evenly (Fig. 16).

Discrepancies between transaction inputs and outputs become even clearer when considering conditional frequencies of transaction categories (Fig. 17, 18). The frequency of separable transactions remains relatively steady at ≈10% until the majority of transactions abruptly becomes intractable (at about 20 inputs). When the number of transaction outputs increases, the increase in the ratio of intractable transactions is smoother; on the other hand, the ratio of simple transactions decreases more abruptly. As plots show, the majority of intractable transactions is likely to be ambiguous (see also Fig. 16). Hence, the amount of mixing shared send transactions among all bitcoin transactions is likely to be close to $(9.6\% + 6.3\% + 2.5\%) \cdot 13.5\% \approx 2.5\%$.

4 Conclusion

Shared send mixing is one of the major types of anonymization techniques in the Bitcoin network. Correspondingly, the problem of untangling shared send transactions (i.e., discovering value flows within the transaction) has great practical importance. In this paper, shared send mixing is formalized using graph notation. By proving NP-completeness of ambiguous transaction detection problem, we show that this measure to obfuscate transaction history is theoretically effective. However, both limitations on the blockchain space and presence of off-chain data can reduce the effectiveness of shared send mixing.

Our computational experiments show that detection and analysis of shared send mixers is possible in real time for the most of bitcoin transactions. We also discover that mixing transactions occur quite often on the Bitcoin Blockchain; by our estimations, they constitute about 2.5% of all bitcoin transactions. Interestingly, about half of these transactions are able to be untangled. Namely, they can be uniquely split into two or more sub-transactions, allowing for the restoration of relationships among addresses referenced in the transaction.

Together with other data mining tools, analysis of shared send transactions forms the analytical backend for Bitfury's Crystal Blockchain – a web service for blockchain investigations and analysis. The first public release of the service is scheduled soon after the publication of the present paper.

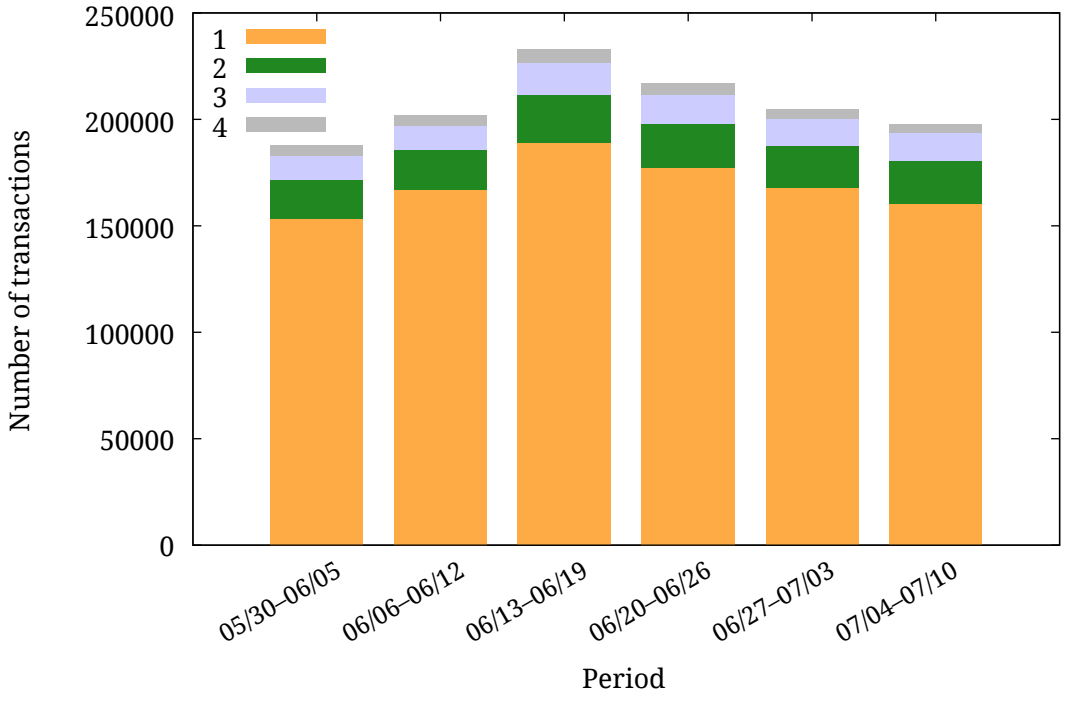

Figure 12: Distribution of shared send transactions among categories from Definition 9, grouped by weeks in the UTC time zone. The following transaction categories are depicted: 1 – simple; 2 – separable; 3 – ambiguous; 4 – intractable.

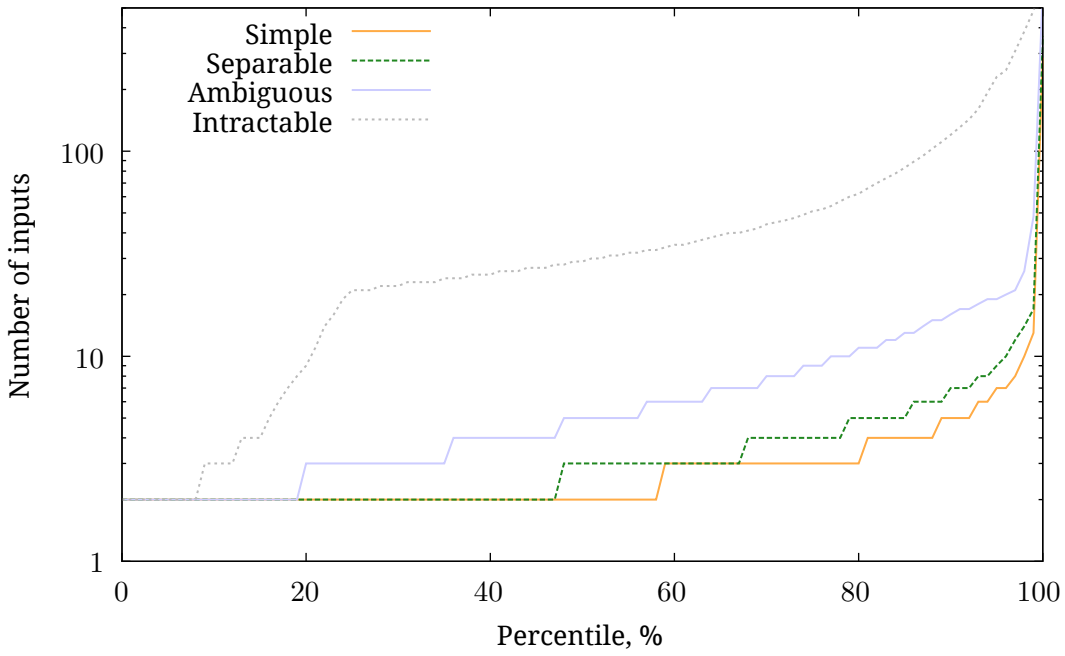

Figure 13: Percent point function for the number of inputs in shared send transactions

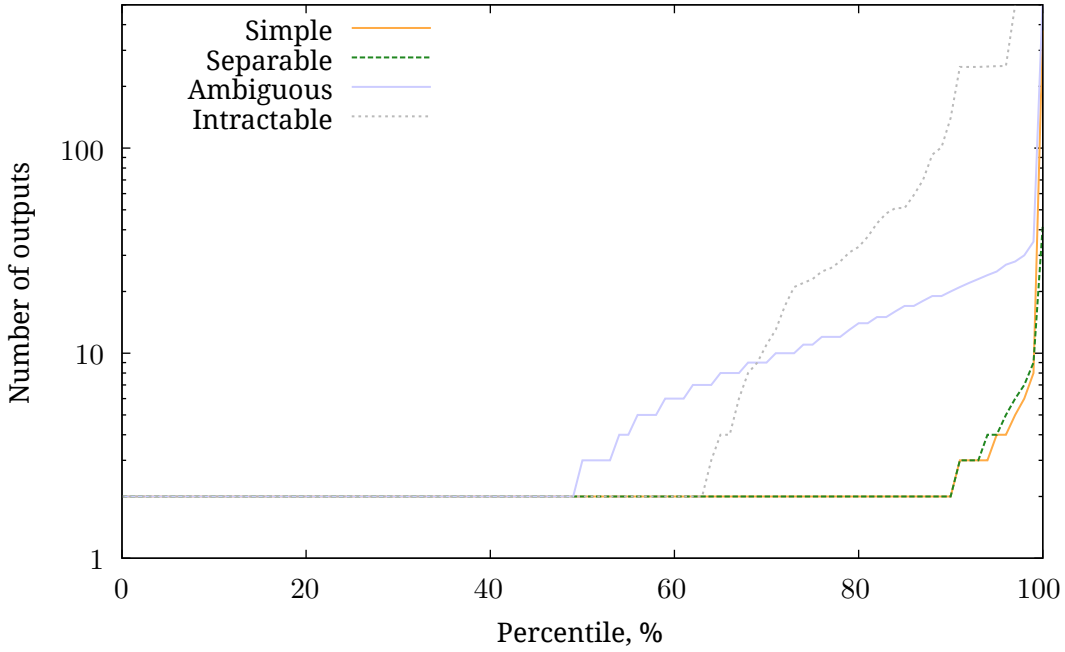

Figure 14: Percent point function for the number of outputs in shared send transactions

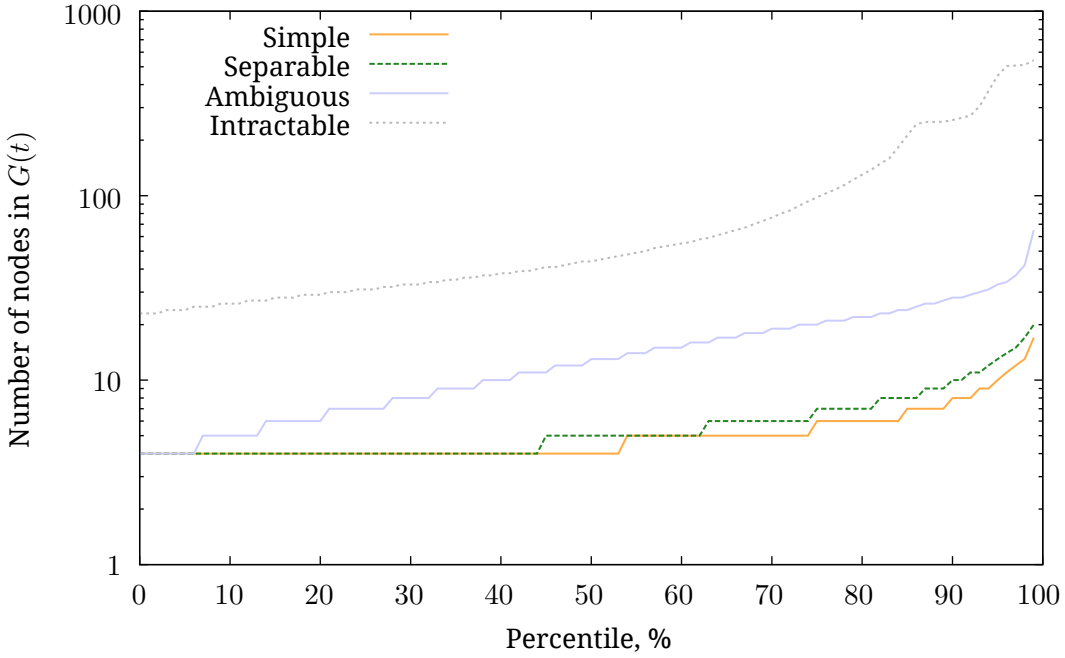

Figure 15: Percent point function for the total number of inputs and outputs in shared send transactions

Table 1: The median and moment-related statistics for the number of inputs, outputs, and the sum of both for the 4 categories of shared send transactions

Characteristic	Transaction type	Statistic			
		median	mean	st. dev.	skewness
Inputs	simple	2	3.0	2.6	16.9
	separable	3	3.8	4.2	15.1
	ambiguous	5	7.7	12.1	13.3
	intractable	29	54.9	85.5	3.7
Outputs	simple	2	2.3	1.5	36.8
	separable	2	2.3	1.4	6.7
	ambiguous	3	7.8	9.6	9.1
	intractable	2	47.0	137.0	7.5
Nodes	simple	4	5.3	3.0	15.7
	separable	5	6.1	4.4	13.3
	ambiguous	13	15.5	14.4	9.8
	intractable	44	101.8	148.6	5.6

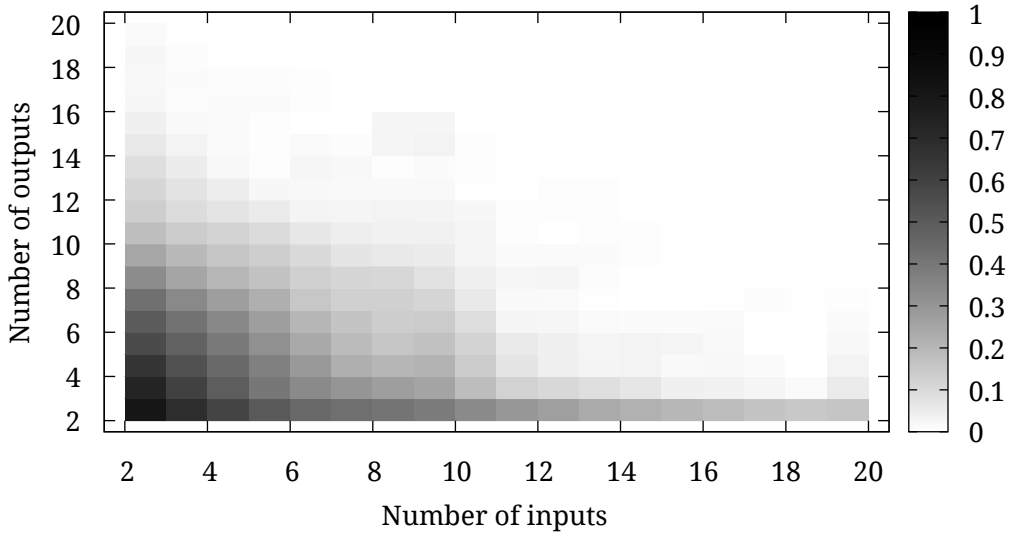

Figure 16: Fraction of separable transactions in tangled transactions. Note that the scale is chosen in such a way that the amount of intractable transactions is minimal, so the fraction can be reliably estimated based on the numbers of separable and ambiguous transactions.

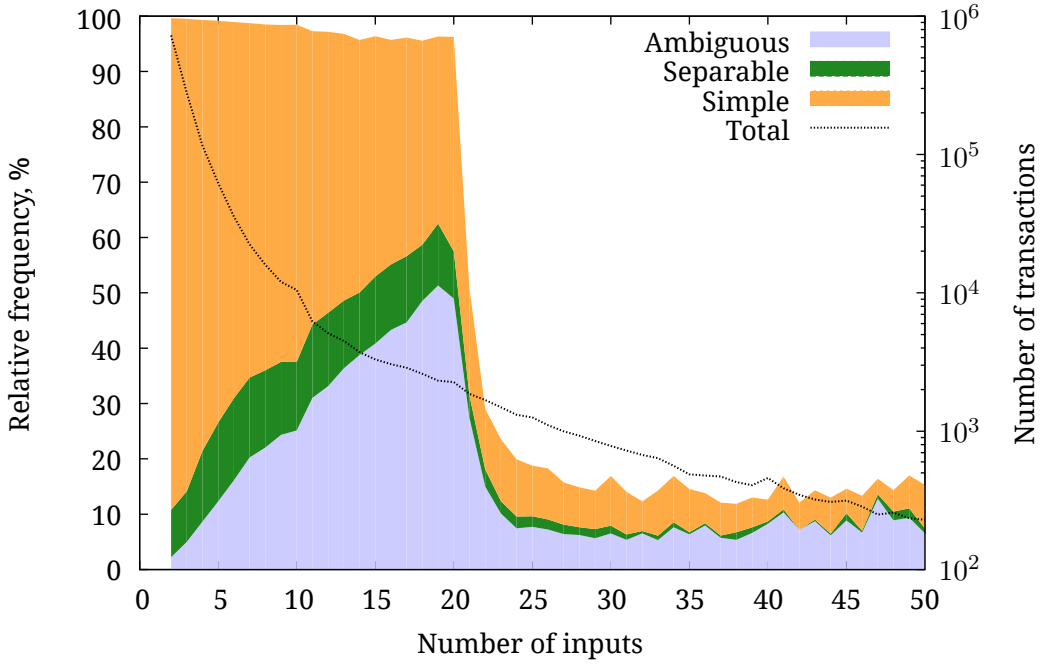

Figure 17: Frequencies of various categories of shared send transactions conditioned on the number of transaction inputs. The blank space corresponds to intractable transactions. Also displayed is the total number of shared send transactions versus the number of inputs (plotted with the right y scale).

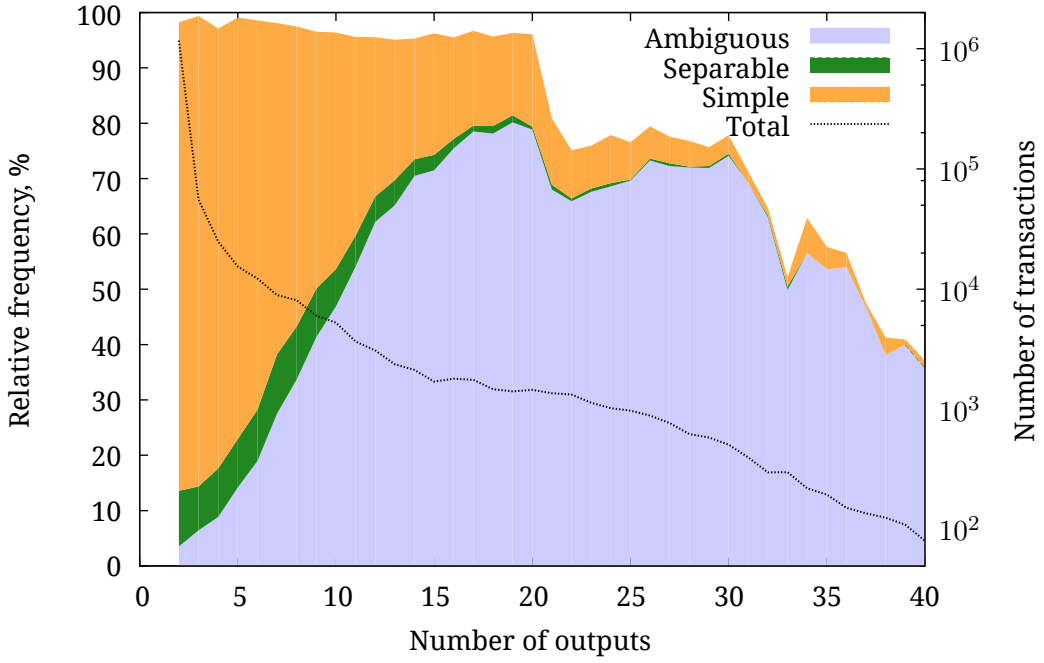

Figure 18: Frequencies of various categories of shared send transactions conditioned on the number of transaction outputs. The blank space corresponds to intractable transactions. Also displayed is the total number of shared send transactions versus the number of outputs (plotted with the right y scale).

References

[1] *Satoshi Nakamoto* (2008). Bitcoin: a peer-to-peer electronic cash system
URL: https://bitcoin.org/bitcoin.pdf

[2] *QingChun ShenTu and JianPing Yu* (2015). Research on anonymization and de-anonymization in the Bitcoin system
arXiv:1510.07782

[3] *Gregory Maxwell* (2013). CoinJoin: Bitcoin privacy for the real world
URL: https://bitcointalk.org/index.php?topic=279249.0

[4] *Fergal Reid, Martin Harrigan* (2011). An analysis of anonymity in the Bitcoin system. In: Proc. 3rd IEEE International Conference on Privacy, Security, Risk and Trust and on Social Computing, SocialCom/PASSAT '11, pp. 1318–1326
doi:10.1007/978-1-4614-4139-7_10

[5] *Simon Barber, Xavier Boyen, Elaine Shi, Ersin Uzun* (2012). Bitter to better – how to make Bitcoin a better currency. In: Proc. 16th International Conference on Financial Cryptography and Data Security, FC '12, pp. 399–414
doi:10.1007/978-3-642-32946-3_29

[6] *Dorit Ron, Adi Shamir* (2012). Quantitative analysis of the full bitcoin transaction graph. Cryptology ePrint Archive, Report 2012/584
URL: http://eprint.iacr.org/2012/584

[7] *Sarah Meiklejohn, Marjori Pomarole, Grant Jordan et al.* (2013). A fistful of Bitcoins: characterizing payments among men with no names. In: Proc. 2013 Internet Measurement Conference, IMC '13, pp. 127–140
doi:10.1145/2504730.2504747

[8] *Michele Spagnuolo, Federico Maggi, Stefano Zanero* (2014). BitIodine: extracting intelligence from the Bitcoin network. In: Proc. 18th International Conference on Financial Cryptography and Data Security, FC '14, pp. 457–468
doi:10.1007/978-3-662-45472-5_29

[9] *Malte Möser, Rainer Böhme, Dominic Breuker* (2013). An inquiry into money laundering tools in the Bitcoin ecosystem. In: Proc. eCrime Researchers Summit, eCRS '13, pp. 1–14
doi:10.1109/eCRS.2013.6805780

[10] *Eli Ben-Sasson, Alessandro Chiesa, Christina Garman et al.* (2014). Zerocash: decentralized anonymous payments from Bitcoin (extended version). Cryptology ePrint Archive, Report 2014/349
URL: https://eprint.iacr.org/2014/349

[11] *Gregory Maxwell* (2013). CoinSwap: transaction graph disjoint trustless trading
URL: https://bitcointalk.org/index.php?topic=321228.0

[12] *Joseph Bonneau, Arvind Narayanan, Andrew Miller et al.* (2014) Mixcoin: Anonymity for Bitcoin with accountable mixes. In: Proc. 18th International Conference on Financial Cryptography and Data Security, FC '14, pp. 486–504
doi:10.1007/978-3-662-45472-5_31

[13] *QingChun ShenTu, JianPing Yu* (2015). A blind-mixing scheme for Bitcoin based on an elliptic curve cryptography blind digital signature algorithm
arXiv:1510.05833

[14] *Kristov Atlas* (2014). Weak privacy guarantees for SharedCoin mixing service
URL: http://www.coinjoinsudoku.com/advisory/

[15] *Valery Vavilov* (2016). The BitFury Group announces expansion to full service digital asset technology company
URL: https://medium.com/@valeryvavilov/the-bitfury-group-announces-expansion-to-full-service-digital-asset-technology-company-4ca739fc7712#.19y5egj8j

[16] *Andreas M. Antonopoulos* (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. O'Reilly Media, 298 p. Chapter 8: Mining and consensus
URL: http://chimera.labs.oreilly.com/books/1234000001802/ch08.html

[17] Secp256k1. In: Bitcoin Wiki
URL: https://en.bitcoin.it/wiki/Secp256k1

[18] Multisignature. In: Bitcoin Wiki
URL: https://en.bitcoin.it/wiki/Multisignature

[19] *Vitalik Buterin* (2014). Bitcoin multisig wallet: the future of Bitcoin
URL: https://bitcoinmagazine.com/articles/multisig-future-bitcoin-1394686504

[20] *Donald Knuth* (1992). Two notes on notation. In: American Mathematical Monthly, vol. 99 (5), pp. 403–422 arXiv:math/9205211

[21] *Pieter Wuille* (2013). Hierarchical deterministic wallets (BIP 32)
URL: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[22] *Paul E. Black* (2016). big-O notation. In: Dictionary of Algorithms and Data Structures, Vreda Pieterse and Paul E. Black, eds.
URL: http://www.nist.gov/dads/HTML/bigOnotation.html

[23] *Michael R. Garey, David S. Johnson* (1979). Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., 340 p.

[24] *Travis Oliphant, Pearu Peterson, Eric Jones et al.* (2001–). SciPy: open source scientific tools for Python
URL: http://www.scipy.org/