Anonymous Alone? Measuring Bitcoin's Second-Generation Anonymization Techniques

Malte Möser
Department of Computer Science
Princeton University
Princeton, USA
mmoeser@princeton.edu

Rainer Böhme
Department of Computer Science
University of Innsbruck
Innsbruck, Austria
rainer.boehme@uibk.ac.at

***Abstract*—This paper contributes a systematic account of transaction anonymization techniques that do not require trust in a single entity and support the existing cryptographic currency Bitcoin. It surveys and compares four known techniques, proposes tailored metrics to identify the use of each technique (but not necessarily its users), and presents longitudinal measurements indicating adoption trends and teething troubles. There is a trade-off between the choice of users' preferred protection mechanisms and the risk that pertaining transactions can be singled out, which hurts privacy due to smaller anonymity sets unless a critical mass adopts the mechanism.**

I. Introduction

The anonymity of financial transactions in the most popular cryptographic currency Bitcoin is far from perfect (cf. [2, 13, 20, 26, 29]). As Bitcoin is built around reusable pseudonyms, its users enjoy a degree of privacy only until someone can link their accounts and activities to a real-world identity. In the worst case, it is possible to look up a user's complete transaction history in Bitcoin's public ledger.

As a remedy, many Bitcoin users have started to rely on centralized bitcoin mixers to make tracing their transactions more difficult. Mixers are services that simply swap the coins of different users. Users must trust the mix not to steal their coins and not to keep any logs that would later allow to reverse the mixing (cf. [8]). These shortcomings have lead to research and development of a second generation of anonymization techniques, which use cryptography and advanced protocols to increase privacy while minimizing the amount of trust required in a central party.

However, they do not eliminate trust in the behavior of others entirely. In particular, many second-generation techniques have special characteristics that allow to distinguish their use from normal Bitcoin transactions. This reduces the anonymity set, a fundamental concept in quantifying privacy [27], to the subset of users of a specific technique, which (as we shall show in this work) is a tiny subset of all transactions. Like in anonymous communication systems, achievable anonymity in transaction systems depends on the adoption of the specific techniques by the community [12].

To better understand the degree of anonymity these systems can provide today, we present a first systematic account of second-generation anonymization techniques in Bitcoin, consisting of theoretical considerations and longterm empirical measurements. To this end, we extract and parse more than 139 million Bitcoin transactions from 88 GB of blockchain data and store them in a Neo4j graph database, allowing us to efficiently query and traverse the transaction graph.

The paper is organized as follows. We review four relevant second-generation anonymization techniques in Section III. Then, in Section IV, we propose metrics to identify their traces in the public ledger based on individual characteristics. This allows us to measure the use of these techniques in the wild. We complement the paper with the necessary background in Section II (safe to skip for readers familiar with Bitcoin privacy concepts), a discussion with implications for future research in Section V, and conclude in Section VI.

II. Background

Bitcoin is a decentralized transaction system that combines an append-only data structure, the public ledger known as the "blockchain", a distributed peer-to-peer network and a probabilistic consensus protocol based on proof-of-work [25]. Owning a bitcoin means that the blockchain contains a digital record stating that one's account identifier, generally a public key, has received funds in the past. Value is transferred through transactions, which consist of a set of inputs and a set of outputs. Outputs reassign bitcoins to a new account identifier and inputs are references to previous outputs that are being spent – completely – in the transaction.

To ensure that only a legitimate owner can spend bitcoins, every output specifies redemption conditions. The input referencing it has to provide the required data to fulfill these conditions. In most cases, the output will require that the input provides a digital signature corresponding to a hash of a public key. However, other types of conditions are possible as well. Multisignature transactions require the input to provide a set of signatures corresponding to multiple public keys (e. g., a 2-of-3 multisignature transactions requires signatures corresponding to two public keys out of a list of three). Hash-locked transactions specify a hash value in the output script, and the input must provide the corresponding preimage in order to redeem the coins. Transactions can furthermore set a locktime that renders them invalid until a certain point in time is reached [10]. This allows users to enforce that a series of transactions can only be committed to the blockchain in a specified order.

Everyone can generate public keys autonomously and use them as account identifiers in Bitcoin. Hence, without additional knowledge, it is not possible to identify individual users taking part in a transaction. However, while users may not be identifiable when transacting solely in the Bitcoin system, many real-world uses rely on exchanges or other intermediaries. Unlike the core system, intermediaries are not decentralized and are often subject to regulation, such as conventional anti-money laundering rules or the Know-Your-Customer (KYC) principle [7]. As a result, someone may establish a direct link between (a subset of) a user's Bitcoin accounts and her identity. Once this relation is known, two common heuristics allow an attacker to extend her knowledge from one account identifier to others.

The first technique to group addresses belonging to the same user is to compute the closure of an address [29, 30], also known as multi-input heuristic. A user who controls multiple bitcoins at different addresses may combine them in a single transaction in order to make a larger payment. As a user's funds are often split up into different addresses after conducting a number of transactions, combining the funds from multiple accounts is a strong indicator for common ownership. A second heuristic works by identifying change outputs [2, 20]. Change outputs are necessary because transactions have to spend all value provided in their inputs. In fact, a scan of the blockchain reveals that 79 % of all transactions created until the end of June 2016 have exactly two outputs, of which one is likely the change output whose funds belong to the user who created the transaction. Although change address detection was much easier in the early days of Bitcoin when clients behaved more predictably, there remain a number of ways to distinguish outputs, such as the occurrence of round values, explicit locktimes, use of P2SH addresses, or bugs that leak information (cf. [35]).

Once an attacker has associated a real-world identifier with a set of account identifiers, she can analyze the transaction behavior of those accounts, e. g., identify the recipient addresses of transactions and the amounts transferred. Transaction graph analysis may also help to reveal and break down obfuscation patterns, such as merges and splits or peeling chains (cf. [20]), and thereby deliver a complete picture of a user's transaction history.

To counter these deanonymization techniques, a **first generation** of mixers evolved, using indicative names such as "Bitcoin Fog", "BitLaundry" or "Bitmixer". Most of them collect funds from many different users and then pay back a slightly smaller amount using coins from (supposedly) unrelated transactions. When using these services, the user has to trust them not to steal their coins, not to keep logs that would allow deanonymization at a later point in time, and there is no guarantee that the returned coins are unlinkable to the paid ones. There is evidence that some mixers directly pay out coins provided by the user, presumably due to a lack of liquidity [24]. Most importantly, mixers have a single point of failure and therefore violate a core principle of Bitcoin. (The Mixcoin [8] proposal tries to mitigate this risk using a reputation system, but it has never been implemented.)

A **second generation** of anonymization techniques has evolved that tries to eliminate the single point of failure while being deployable in the current Bitcoin system.[1] These techniques can broadly be categorized into low-level anonymization techniques [3, 16, 17, 36] as well as high-level mixing protocols [5, 15, 32], some of which are partially based on the low-level techniques. As none of these high-level mixes has been fully deployed so far, this work focuses on the low-level techniques. These include Fair Exchange [3] and CoinSwap [17], which allow users to swap coins while ensuring that no party can steal them; CoinJoin [16], which aims at obscuring the flow of funds by combining the payments of multiple users into one transaction, and stealth addresses that aim at preventing an attacker from linking real-world identifier to Bitcoin addresses [36]. A disadvantage of these techniques, however, is that in order to achieve their goals, they use constructions that differ from the standard use of the Bitcoin network and thereby are identifiable on the blockchain. As we are not aware of any systematic account of second-generation anonymization techniques, we set out to analyze these different techniques and provide measurements based on the characteristics identified.

III. Anonymization Techniques

We now review the different second-generation anonymization techniques and discuss their anonymity properties.

A. Fair Exchange

A simple idea to thwart blockchain analysis is to swap one's coins with another Bitcoin user. When Alice spends Bob's coins and Bob uses Alice's coins, a passive attacker interested in Alice's spending behavior will no longer observe her transactions, but will observe Bob's instead. The Fair Exchange protocol by Barber et al. [3] allows such a coin swap between two parties in a trustless way using time- and hash-locked transactions. The idea behind the protocol is that when Bob claims the coins of Alice, he has to reveal a secret that allows Alice to also claim coins from Bob. The protocol (which we explain in a simplified way) is designed in such a way that after running it, either both parties received what they expected or get their funds back.

Protocol: At the first stage of the protocol, Alice and Bob each generate a set of private values a_i and b_i. Then, they engage in a cut-and-choose protocol that will provide Alice with a set of correct hashes $\mathsf{H}(b_i)$ and $\mathsf{H}(a_i+b_i)$. In the second stage (cf. Figure 1), Alice and Bob lock their funds in hash-locked transactions. Bob creates a transaction T_{CB} whose outputs can be spent with either both signatures of Alice and Bob, or just a signature of Alice and any value b_i. To ensure that he gets back his coins if Alice abandons the protocol, Bob also creates a refund transaction T_{RB} with a future locktime t that returns the coins to him. He lets Alice sign the refund transaction

[1] By contrast, another stream of research proposes modifications to the protocol in order to improve privacy [4, 18, 21, 33]. Here we are only interested in measuring deployed systems.

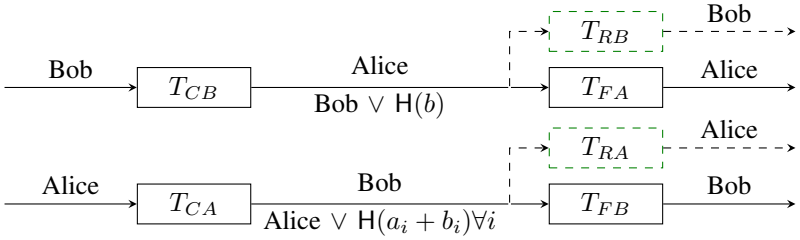


Fig. 1: Transaction structure for a Fair Exchange including unpublished refund transactions (dashed, green)

before he publishes T_{CB}. Alice then creates a similar claim transaction T_{CA} that can be redeemed with a signature of Bob combined with either a signature of Alice or combinations of two values $a_i + b_i : \forall i$. She also creates a refund transaction T_{RA} with a locktime of $t + 1$ and only publishes T_{CA} after receiving a valid signature for T_{RA} from Bob. With this setup, Bob can claim Alice's coins by providing both his signature and $(a_i + b_i) : \forall i$ in the redeeming transaction. Alice, in turn, can compute any valid b_i by subtracting a_i from $a_i + b_i$ and then use just one of the values in order to claim Bob's funds.

Anonymity: The anonymity of Fair Exchange is based on two characteristics. First, Alice and Bob do indeed effectively swap their coins. If they use different key pairs for the claim transactions, there is no direct connection between the transactions in the public ledger. Second, the transactions are not directly linkable via the preimages a and b because the connection is hidden within the hash and the sum of the values. However, the scheme requires two very special output scripts that, while not being directly relatable to each other, clearly reveal the use of the scheme on the blockchain. Furthermore, both claimed transaction outputs are likely to contain roughly the same value. Hence, the size of the anonymity set is limited to those transactions that contain similar redemption conditions and values. Due to the locktime of the refund transactions, claiming the funds must happen shortly after the funding transactions have been published, further restricting the feasible size of the anonymity set. This means that Fair Exchange can only add anonymity if it is used by a large enough set of users with swapping comparable values at roughly the same time.

B. CoinSwap

CoinSwap [17] allows two parties (Alice and Bob) to pay each other without making a direct, traceable transaction. Instead, a third person (Carol) receives coins from Alice and pays Bob with coins that are not related to the ones she received from Alice. Using multiple escrow transactions, this scheme can conduct the swap without requiring trust between the parties. However, if one of the three misbehaves, the swap may only be resolved by using hash-locked transactions that are linkable in the public ledger.

Protocol: To initiate a CoinSwap, Alice and Carol lock their funds in a 2-of-2 multisignature output (cf. Figure 2). Alice's transaction T_0 can only be redeemed with signatures of both Alice and Carol, and Carol's transaction T_1 can only be redeemed with signatures of both Carol and Bob. Then,

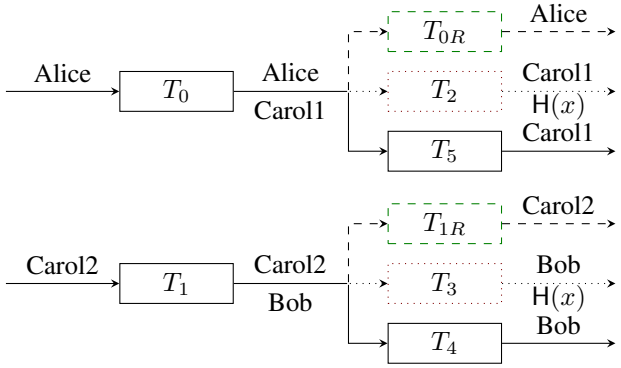


Fig. 2: Transaction structure for a CoinSwap including unpublished refund (dashed, green) and hash-locked (dotted, red) transactions

Carol and Bob create time-locked refund transactions T_{0R} and T_{1R} that can be used to send back the locked funds to Alice and Carol if a party abandons the protocol. The transactions' locktimes must be carefully chosen in order to ensure fairness, i. e., T_{0R} must be locked longer than T_{1R}. After ensuring that T_{0R} and T_{1R} are correctly signed, Alice and Carol publish T_0 and T_1 on the network.

Next, Bob selects a random value x, computes $\mathsf{H}(x)$ and sends this hash value to Alice and Carol. Alice creates a hash-locked transaction T_2 that allows Carol to spend the money locked in T_0 once she knows x. Carol computes a similar transaction T_3 that would allow Bob to redeem the funds in T_1. In order to claim the coins, Bob must publish x, which would allow Carol in turn to claim T_3. However, providing x in both transactions would allow to link these transactions in the public ledger. These transactions are merely to claim funds if a party misbehaves and should not be published otherwise. Once Bob has received T_3 from Carol, he can be certain that he will receive his funds and will then give x to Carol. Without learning x, Carol should not continue the protocol as she has no way to pull her funds from Alice once she paid Bob.

When all parties are committed to the swap, Carol and Alice create the final transactions to transfer funds to Bob and Carol. Carol creates T_4 and sends it to Bob, who then signs the transaction and publishes it to the network. Once confirmed, Alice creates T_5 and sends it to Carol, who in turn signs the transaction and publishes it.

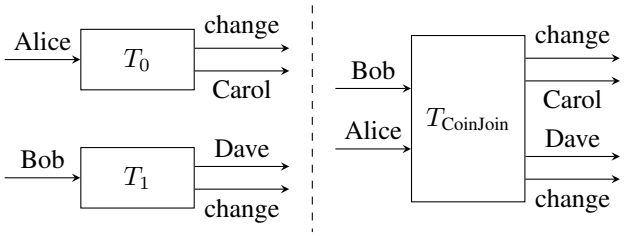


Fig. 3: Two or more individual transactions (left) can be combined in a single CoinJoin transaction (right)

Anonymity: The anonymity of CoinSwap relies on the cooperation of the parties involved to avoid the use of the hash-locked transactions, as well as on the indistinguishability of the transactions from normal 2-of-2 multisignature transactions. Because of the time-locked refund transactions, the anonymity set is restricted to 2-of-2 multisignature transactions within a certain time frame. Although the transactions are not directly linkable in the blockchain by account identifiers, both T_0 and T_1 may be linkable by the amount they transfer, as Carol likely wants to receive the same amount from Alice that she pays to Bob. Finally, Carol must not combine funds that are relatable to account identifiers of Carol1 with those of Carol2, as this would allow linking both ends in the transaction graph.

C. CoinJoin

In CoinJoin, instead of having multiple users each sending a single transaction, they create one joint transaction (cf. Figure 3) [16]. As each transaction input has to provide a valid signature independently of the other inputs, and every signature is computed over the whole transaction, and is thereby valid for only one exact combination of inputs and outputs, participants in a CoinJoin can rest assured that neither of the other participants is in a position to steal their funds. An attacker cannot easily see which inputs and outputs belong together. In case of widespread adoption, CoinJoin could reduce the applicability of the multi-input deanonymization heuristic. It also increases the difficulty of deriving to which address a user sends her funds. The anonymity provided by CoinJoin hence depends on the indistinguishability of these transactions from normal transactions (preventing an attacker from excluding CoinJoin transactions from the multi-input heuristic) and the hardness to infer the true subsets of inputs and outputs [19].

It is easy to demonstrate that only combining the inputs and outputs of multiple transactions does not necessarily increase anonymity. When the amounts of inputs and outputs leak sufficient information, matching those values allows an attacker to infer which inputs and outputs belong together [19]. For example, Alice might want to send 10 bitcoins to Carol and has an input with a value of 11 BTC. Bob wants to send 0.5 BTC to Dave and therefore combines two inputs with a value of 0.4 BTC each. It is easy to infer subsets in the resulting CoinJoin transactions because the values match up nicely (cf. Figure 4a). Now one could apply the multi-input heuristic only to the inputs in each subset.

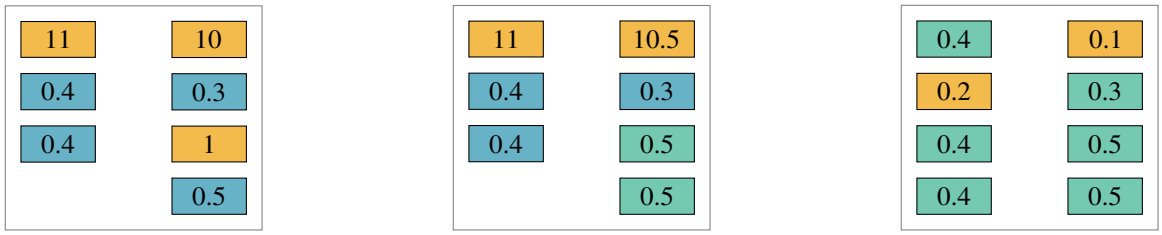
Maxwell [16] recommends a uniform output value for the spending output because this creates an anonymity set of all potential recipients. If we reduce the value of the spending output in our example to 0.5 BTC (cf. Figure 4b), it is no longer possible to derive one exact combination of subsets for inputs and outputs. Instead, either 0.5 BTC output could belong to the orange or the blue subsets. However, it is still reasonable to assume that both inputs of 0.4 BTC have been combined to produce one of the outputs spending 0.5 BTC, hence we can still determine subsets for the inputs and infer the change addresses in the outputs. The uniform output size has, in fact, made it easier to detect the change addresses as one can rule out the outputs with equal values.

In practice, subset matching must take into account that transaction fees reduce the value of one or multiple output subsets. To make this fuzzy matching even more difficult, one can increase the number of participants and also align the values of the inputs (ideally with a maximum difference in the range of the transaction fee). If Alice uses two small inputs with values of 0.2 BTC and 0.4 BTC instead of her large 11 BTC input (cf. Figure 4c), there are ten possible combinations to partition the inputs and outputs into matching sets. However, when taking into account that the transaction at hand is a "strict" CoinJoin transaction, meaning that with n participants there will be n outputs of uniform size, we can discard solutions where not every output subset includes such an output. In the example, this lets us infer that the 0.1 BTC output must be the change of the 0.2 BTC input.

The original proposal claims that CoinJoin will render the multi-input heuristic obsolete because a CoinJoin "is externally indistinguishable from a transaction created through conventional use" [16]. However, if participants agree on a common output size, they form a distinct characteristic for the final transaction that allows an attacker to distinguish between CoinJoins and normal transactions. Furthermore, the difficulty of finding matching subsets increases with the number of participants. But in turn, this also makes the transaction stand out more in comparison to all other transactions. As of June 2016, at least 90 % of all transactions have two or fewer outputs. This means that the indistinguishability of a CoinJoin transaction is inversely related to the difficulty of finding subsets as well as to the anonymity set given by the number of potential recipients.

D. Stealth Addresses

Stealth addresses [36] solve the problem that whenever someone publicly posts a Bitcoin address to receive money, anybody can look up the payments made to this particular address. Even if an address is not published, but reused for different recipients, multiple transaction can be associated with the same address. Since offering unique Bitcoin addresses to all potential senders can be challenging, stealth addresses enable a recipient to publish a single static identifier. This identifier allows each sender to derive a unique Bitcoin address that cannot be associated with the public identifier. The recipient of a transaction can compute the corresponding private key



(a) Distinct subsets **(b)** Possible subsets **(c)** Noisy subsets

Fig. 4: Examples for subsets of CoinJoin transactions

belonging to the derived Bitcoin address based on additional data that is included alongside the normal outputs using the `OP_RETURN` statement.

Protocol: Stealth addresses are based on the Diffie–Hellman key exchange on elliptic curves. In the following, we present the protocol as described in [36] and [9]. When Bob wants to receive money whilst enjoying recipient anonymity, he first generates an ECDSA key pair $Q = d \cdot G$ (G is a publicly known generator) and then publishes the public key Q as static identifier. Alice now generates her own temporary key pair $P = e \cdot G$ and computes a shared secret $c = \mathsf{H}(e \cdot Q)$. She then uses c to derive a point $Q' = Q + c \cdot G$, converts Q' into a Bitcoin address and sends bitcoins to it. In the transaction she also includes the previously generated point P in an `OP_RETURN` output. For each transaction on the blockchain that contains a possible P, Bob computes $c = \mathsf{H}(d \cdot P)$ and checks whether this allows him to derive a valid point $Q' = (d + c) \cdot G$. Then he can spend the funds at Q' by computing the matching private key $d' = d + c$.

Anonymity: Stealth addresses aim at preventing an attacker from tying external information to an address that allow to identify a Bitcoin user (see [35] for examples). If implemented without further obfuscation, they are easily detectable on the blockchain due to P being embedded in the `OP_RETURN` output. However, while it is possible to detect the usage of a stealth address, it is infeasible for a attacker to reconstruct the original stealth address Q (or address pair (Q, R)). Hence, the recipient enjoys recipient anonymity.

IV. Measurements

We now propose metrics for quantifying the use of the anonymization techniques and present measurements using blockchain data extracted from the block files of the Bitcoin core client. Our dataset includes all transactions until end of June 2016 (block height 418,722).Whenever we report transaction volumes, we aggregate data to *periods* of 1008 consecutive blocks (about one week). The number 1008 corresponds to exactly one half of a difficulty adjustment cycle in the Bitcoin protocol and has become a common unit of analysis for longitudinal studies [23].

A. Fair Exchange

Recall that the Fair Exchange protocol uses two special redemption conditions. The first requires a set of values $(a_i + b_i)$, the second any value b_i. The resulting output scripts hence have a characteristic structure including an if–then–else block [3].

While normal clients typically consider these scripts non-standard and do not relay these transactions, the Pay-to-Script-Hash transaction type allows to use almost any type of script as it reveals the full script only when spending an output that has already been included in the blockchain. To identify Fair Exchange transactions, we therefore analyze all non-typical transaction scripts in transaction outputs as well as the scripts provided in inputs that redeem Pay-to-Script-Hash outputs. In total, this gives us 616,693 scripts.

After sanitizing by removing all data elements and extracting the top 100 non-standard scripts, we evaluate each of them manually. The most frequent non-standard script types are empty `OP_RETURN` scripts (288,552 times) and scripts containing `OP_FALSE` (219,319 times), but we also see a few thousand strange scripts making use of multisignature constructs. However, we are not able to find any script in this set that matches the format required by Fair Exchange. As the least popular script in this list has not been used more than seven times, we conclude that Fair Exchange has not seen any practical use to this day.

B. CoinSwap

Each CoinSwap uses two multisignature transactions that require 2-of-2 signatures. Already in mid-2015 we observe about 15 to 25 such multisignature transactions per block (we report these measurements in Section A), which should be sufficient for a potential use of CoinSwap. Therefore we have developed five criteria to estimate the number of *potential* CoinSwaps:

1) The protocol must finish before the locktime of the refund transactions elapses. This prevents situations where Carol payed Bob, but Alice reclaims her funds using the refund transaction. The protocol does not specify a value for the locktime, thus we assume that the redeeming transaction must spend the output within a timespan of 30 blocks (about 5 hours). This gives all parties more than enough time to communicate their transactions and get them confirmed while ensuring that funds are not locked up for too long.
2) Our second criterium is the value of the output. If Alice pays Carol 1 BTC, there is no reason for Carol to pay 2 BTC to Bob. To allow for Alice reimbursing Carol's

transaction fees, we define that outputs may match if their values differ up to 0.002 BTC, twice a fee of 0.001 BTC, which was common in 2013 and significantly lower thereafter [23].

3) We require that the funding transactions for the 2-of-2 multisignature outputs must both have taken place before one of them was spent again. This follows trivially from the protocol as Alice, for example, would not pay Carol before being certain that Carol also pays Bob.
4) We exclude any transactions that include an `OP_RETURN` output as those hint at non-currency transactions (such as Colored Coins, cf. [31]).
5) CoinSwap only provides anonymity if the two outputs are not connected in the transaction graph. Therefore, we ensure that any two outputs we compare are unconnected by traversing the transaction graph backwards and checking that the other output is not referenced in any of the inputs of the preceding transactions.

Our proposed estimation method works as follows: First, we extract all 2-of-2 multisignature outputs that have been spent within 30 blocks in chronological order. We create a pruned set by removing all outputs belonging to transactions that have three or more elements in the (original unpruned) set. The rationale is that while having more than one 2-of-2 multisignature output in a transaction is fully compatible with the scheme and might even help to hide the swap, there exist some transactions that contain tens or hundreds of multisignature outputs. As we cannot find a good reason to build CoinSwaps of that size, we prefer to eliminate these elements which would otherwise appear as outliers or spikes. Finally, we check for each output in the pruned set whether there is another output that fulfills the conditions outlined above within a time frame of 30 blocks ahead of the block height. The number of matches gives us an upper bound for the number of potential CoinSwaps (i. e., it may serve as an upper bound for the size of the anonymity set). It is possible, but computationally more demanding, to tighten the bound by finding the combinatorial solution.

Figure 5 shows the result of our estimation. It reports the number of outputs for which we find a matching output (i. e., a potential CoinSwap) over time. There are almost no matches before 2015, when the general usage of 2-of-2 multisignatures has also been low. In 2015 there are on average only a few matches per block. However, similar to the rise of 2-of-2 multisignature outputs in 2016 (cf. Section A) we observe a strong increase in potential CoinSwap outputs in the first half of 2016, to up to 50 matches per block.

In order to provide anonymity, these outputs must also match the value a user wants to anonymize. The blue line in Figure 6 shows the distribution of the outputs' values. 28 % of the outputs have values below 0.01 BTC, and 59 % have values below 0.1 BTC, which corresponds to at most a mid two-figure USD amount at current market prices. This might be sufficient for smaller trades, but is too low if people want to anonymize larger transfers. The red line in Figure 5 shows all potential CoinSwaps above 0.01 BTC (6 USD). We see

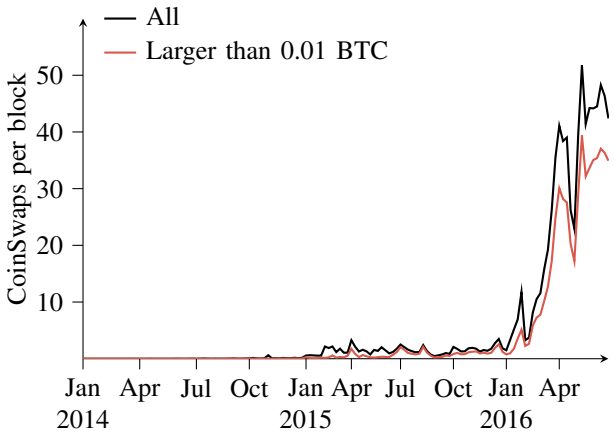


Fig. 5: Upper limit of potential CoinSwaps per block

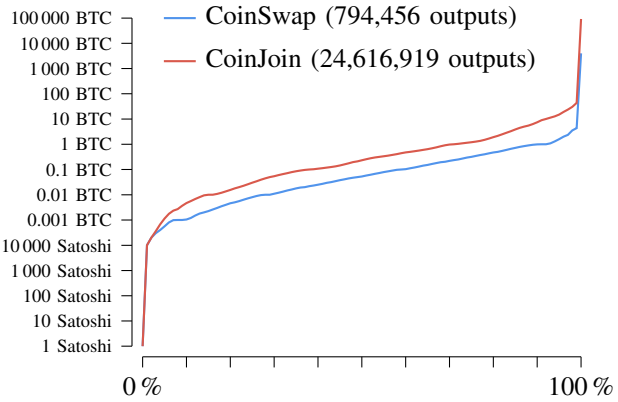


Fig. 6: Cumulative distribution of CoinSwap and CoinJoin output sizes (logarithmic scale)

that removing small outputs reduces the anonymity set only slightly. Furthermore, larger sums can always be split into multiple swaps, thereby increasing the anonymity set at the cost of higher transaction fees.

As we are not aware of any services that simplify the creation of CoinSwap transactions, we assume that most of the 2-of-2 multisignature outputs are the result of normal use (e. g., usage of safe wallets or micropayment channels [14]) and do not represent CoinSwaps. However, our measurements show that there exists a large enough anonymity set for potential CoinSwaps to hide.

C. CoinJoin

To measure CoinJoin transactions, we first define them using a set of rules. As the minimal number of participants in a CoinJoin is two, and if we exclude the special case where a party spends exactly the amount she has available at one of her inputs, a CoinJoin can be any transaction that has at least four outputs: two for the actual spending and two for the change. If each party claims two outputs, the number of inputs must amount to at least half the number of outputs.

To rule out cases where a user combines multiple outputs that belong to the same address or splits up her funds and sends different values to the same address, we do not simply look at the number of inputs and outputs but instead require that the number of unique addresses matches those thresholds. Next, we filter out transactions that send to addresses starting with `1dice` as those are (in most cases) deposit addresses for the gambling service SatoshiDice and known to cause false positives [19]. Again, we exclude non-currency transactions, i. e., those that contain an `OP_RETURN` output.

By participating in a few CoinJoin transactions ourselves, we discover that different services interpret the recommendation for uniform sizes (cf. Section III-C) differently. While JoinMarket[2] ensures that its spending outputs all have the same size (cf. [22]), Shared Coin[3] produced transactions that do not have this property. We are especially interested in the share of these "strict" CoinJoin transactions where each spend has the same value and therefore provide an additional measurement of those.

Figure 7 shows the number of transactions meeting these rules over time. For comparison, we also plot the amount of "large" transactions, i. e., all transactions with more than two inputs and more than four outputs. The number of potential CoinJoin transactions grew from zero per block to relatively stable values between 10 and 15 transactions per block in 2014 and 2015. Observe that only after the technique was proposed in August 2013, the number of matching transactions went up. In comparison, the number of all large transactions is between five to ten transactions per block higher. The number of "strict" CoinJoin transactions is rather low, mostly between one and two transaction per block.

Inspecting the distribution of output values of CoinJoin transactions, we see a slight tendency towards "round" output sizes (cf. plateaus in Figure 6). Output sizes such as 0.01 BTC or 1 BTC are chosen more often than unround values.

Given that we see such a large share of potential CoinJoin transactions, it would be interesting to further determine how many of them are indeed CoinJoin transactions and how many might simply look like they are. Additional information available on blockchain.info helps to identify transactions that were created using Shared Coin, which was probably the most popular CoinJoin service in the past years. Because Shared Coin was integrated into the online wallet of blockchain.info, transactions created by the service should be relayed by blockchain.info itself first. We sample 10,000 transactions between August 30, 2013 and June 30, 2015 from the transactions which we identified as potential CoinJoin transactions and inspect the IP address from which blockchain.info has first heard of a transaction by parsing publicly available data from their website.

Figure 8 shows the share of CoinJoin transactions originating from blockchain.info over time. In total, 57 % of our sampled transactions can likely be attributed to Shared Coin. The graph exhibits a few interesting shifts. Shared Coin was available to users of the online wallet since November 2013 [6]. The increase in the share can hence be explained with the release of the service. In January 2015 the share of Shared Coin transactions dropped by almost two thirds with no significant change in the number of CoinJoin transactions. We conjecture that this is due to a change in the implementation that added support for additional API endpoints, increasing the number of servers that push the transaction into the network [28]. When we compute the share of Shared Coin transactions for our CoinJoin transactions created with Shared Coin in mid-2015, only a third of those transactions are identified as being relayed by blockchain.info. This number matches the estimated share over the course of 2015 surprisingly well.

D. Stealth Addresses

Stealth payments can be detected by searching for raw public keys (or an extended format used by Dark Wallet [9]) in `OP_RETURN` outputs. Figure 9 shows the total number of detected stealth outputs per period. The first stealth transaction[4] took place in February 2014, even before the release of version 0.9.0 of Bitcoin Core, which added transactions with `OP_RETURN` outputs to the list of standard transaction types. The overall usage of stealth addresses is extremely low for the whole time frame, between zero and 60 transactions in each period. The volume peaked to a maximum of 180 transactions in the second quarter of 2015 for no identifiable reason. Perhaps somebody tested stealth addresses as part of a wallet implementation.

A reason for the low adoption rate of stealth addresses could be the lack of a common standard. This, for example, delayed and hindered the integration of stealth addresses into the popular wallet software Electrum (cf. [1]). While stealth addresses provide a unique and pretty elegant opportunity to increase the privacy of users, they currently seem to have no practical relevance.

V. Discussion

An essential property of anonymity is that it cannot be achieved in isolation. Only when a group of entities shares a set of characteristics, they form an anonymity set in which the individual enjoys a level of indistinguishability [27]. For Bitcoin, this yields a fundamental problem: while individual transactions are already highly distinguishable due to their unique transaction histories, anonymization techniques add further complexity to the system, and individual techniques hurt each other's effectiveness due to their direct competition. Complexity originates from the need to design protocols that avoid single points of failures, which in turn exhibit special properties that can be exploited to reduce their anonymity set. It furthermore limits the rate of adoption when users need to install specialized software, and opens the doors for a multitude of operational risks. Competition and incompatibility among different anonymization techniques hurt anonymity in that they partition the set of users and thereby reduce the

[2]https://github.com/JoinMarket-Org/joinmarket, retrieved on 2016-07-26

[3]This services was available at http://sharedcoin.com but has been discontinued in the first half of 2016.

[4]ID: `63e75e43de21b73d7eb0220ce44dcfa5fc7717a8decebb254b31ef13047fa518`

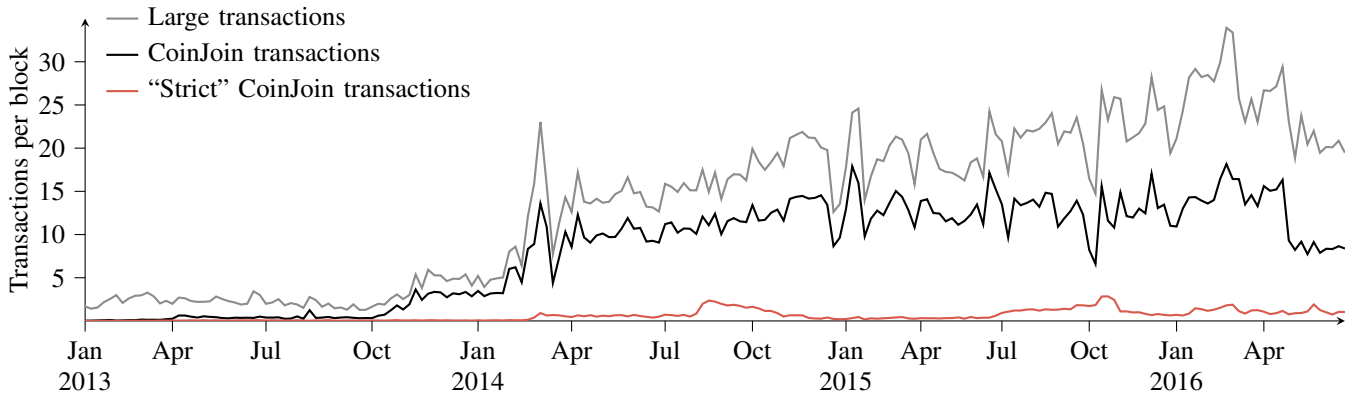


Fig. 7: Number of CoinJoin transactions per block

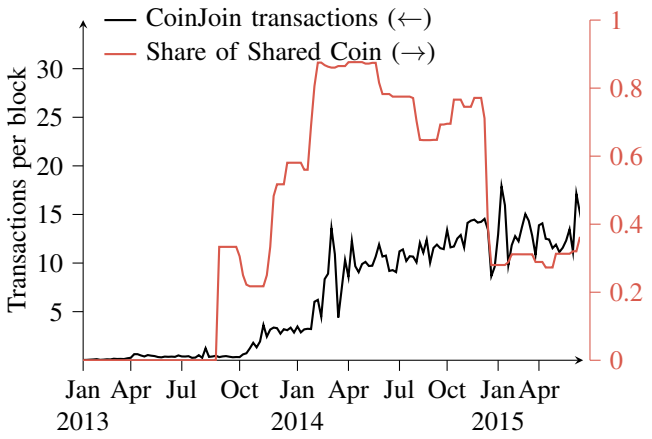


Fig. 8: Estimated share of CoinJoin transactions per block attributed to Shared Coin

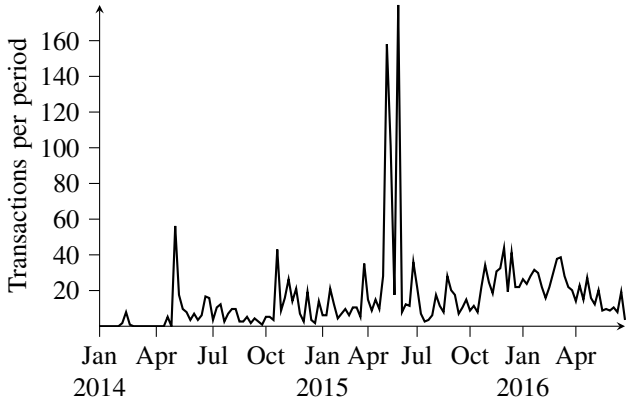


Fig. 9: Total number of stealth transactions per 1008 block period

size of the anonymity set further. As a result, and standing against fundamental principles of Bitcoin, maintainers of the system should be aware that giving users the choice of which anonymization technique to use may not be in their own best interest.

A major challenge in the design of anonymizing techniques is the quantification of the degree of anonymity they offer. Established notions of and metrics for anonymity (e. g., based on entropy [11, 34]) and unlinkability in related subfields do not directly apply to cryptographic currencies where coins are ascribed value *because* they are verifiably linkable to their transaction history. Moreover, Bitcoin transactions cannot be analyzed in isolation. The users' anonymity is not only influenced by the individual characteristics of their transactions, but also the unique context in the overall transaction graph. This yields interesting novel trade-offs. CoinJoin transactions, for example, may provide low "local" anonymity (i. e., being less resistant to subset matching) but thereby better improve overall anonymity for their higher indistinguishability from "normal" transaction behavior. Finding new models to study, quantify and measure these different aspects, both with regard to local characteristics of a transaction and its overall context in the transaction graph, is an interesting avenue for future research.

Besides developing better notions of anonymity for distributed transaction systems, there remains the potential to conduct more specific analyses of the techniques presented. One possibility would be to identify aborted CoinSwaps based on hash-locked or time-locked transactions in order to quantify the ability of participants to find reliable mixing partners or the utility of reducing the trust needed for mixing. Blockchain-based information could also be enriched with information from monitoring the Bitcoin peer-to-peer network, e. g., to correlate specific transactions with timing or location information relatable to individual participants. A major limitation of such measurement studies, including our own, is, however, the ability to verify the findings. Due to a lack of implementations for most techniques there is little ground truth data available. And trying to study especially those transactions that aim at concealing spending behavior also does not make verification easier.

In the end, successful anonymization does not only rely on secure protocols and correct implementations of those, but must also address the trust issues that remain in second generation anonymization techniques. While the techniques presented remove the need to entrust a single third party

with one's coins, they do not necessarily protect against other participants in the protocol from revealing private information that can reduce one's own privacy, both unintentionally and intentionally. Notions of reputation or sanctions (cf. [5, 8]) can help to mitigate these issues and should be an integral part of high-level protocols built upon these techniques.

VI. Conclusion

This paper contributes a first systematic account of second-generation anonymizing techniques in Bitcoin that do not put users' funds at risk and can be used with the current system. The range of available techniques is not extensive, but offers enough to tackle different aspects of deanonymization. We propose metrics to quantify the use of these techniques and present measurements. We find that CoinSwap has a potentially large anonymity set, chiefly because it blends in with normal 2-of-2 multisignature outputs. Stealth addresses are promising to increase the privacy of users who need publicly-known addresses, but a lack of consensus on their technical details is responsible for its low usage. CoinJoin is the most popular technique so far. It can provide different variants of anonymity, but even this technique is not yet widely supported.

To move research on the state of anonymity in Bitcoin forward, we need a better assessment of the overall degree of anonymity that not only takes single transactions but their contexts into account, as well as timing-based and inference attacks. If such work continues the tradition of finding more effective deanonymization approaches, the maintainers of the system may want to consider tackling the issue more directly in the protocol.

VII. Acknowledgments

Parts of this work were funded by the German Bundesministerium für Bildung und Forschung (BMBF) under grant agreement No. 13N13505.

References

[1] *Added ability to send to stealth addresses.* 2014. URL: https://github.com/spesmilo/electrum/pull/817 (visited on 2016-06-26).

[2] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. "Evaluating User Privacy in Bitcoin." In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Vol. 7859. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2013, pp. 34–51.

[3] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. "Bitter to Better — How to Make Bitcoin a Better Currency". In: *Financial Cryptography*. Vol. 7397. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 399–414.

[4] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized Anonymous Payments from Bitcoin". In: *Proceedings of the 2014 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2014, pp. 459–474.

[5] George Bissias, A. Pinar Ozisik, Brian N. Levine, and Marc Liberatore. "Sybil-Resistant Mixing for Bitcoin". In: *Proceedings of the 13th ACM Workshop on Workshop on Privacy in the Electronic Society*. ACM, 2014.

[6] Blockchain. *Shared Coin Announcement Tweet*. 2013. URL: https://twitter.com/blockchain/status/402224010492006400 (visited on 2016-06-26).

[7] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. "Bitcoin: Economics, Technology, and Governance". In: *Journal of Economic Perspectives* 29.2 (2015), pp. 213–238.

[8] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. "Mixcoin: Anonymity for Bitcoin with Accountable Mixes". In: *Financial Cryptography and Data Security*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Vol. 8437. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 486–504.

[9] *DarkWallet/Stealth*. URL: https://wiki.unsystem.net/en/index.php?title=DarkWallet/Stealth&oldid=3746 (visited on 2016-06-26).

[10] *Developer Guide*. URL: https://bitcoin.org/en/developer-guide (visited on 2015-08-11).

[11] Claudia Diaz, Stefaan Seys, Joris Claessens, and Bart Preneel. "Towards Measuring Anonymity". In: *Privacy Enhancing Technologies*. Ed. by Roger Dingledine and Paul Syverson. Vol. 2482. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2003, pp. 54–68.

[12] Roger Dingledine and Nick Mathewson. "Anonymity Loves Company: Usability and the Network Effect". In: *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*. Ed. by Ross Anderson. Cambridge, UK, June 2006.

[13] Martin Harrigan and Christoph Fretter. *The Unreasonable Effectiveness of Address Clustering*. 2016.

[14] Mike Hearn and Jeremy Spilman. *Bitcoin contracts*. URL: https://en.bitcoin.it/%20wiki/Contracts (visited on 2015-10-08).

[15] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. *TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub*. 2016.

[16] Gregory Maxwell. *CoinJoin: Bitcoin Privacy for the Real World*. 2013. URL: https://bitcointalk.org/index.php?topic=279249.0 (visited on 2016-06-26).

[17] Gregory Maxwell. *CoinSwap: Transaction graph disjoint trustless trading*. 2013. URL: https://bitcointalk.org/index.php?topic=321228.0 (visited on 2016-06-26).

[18] Gregory Maxwell. *Confidential Transactions*. 2015. URL: https://people.xiph.org/~greg/confidential_values.txt (visited on 2015-07-06).

[19] Sarah Meiklejohn and Claudio Orlandi. "Privacy-Enhancing Overlays in Bitcoin". In: *Financial Cryptography and Data Security, 2nd Workshop on BITCOIN Research*. Ed. by Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff. Vol. 8976. Lecture

Notes in Computer Science. Berlin Heidelberg: Springer, 2015, pp. 127–141.

[20] Sarah Meiklejohn, Marjoir Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names". In: *USENIX ;login:* 38.6 (2013).

[21] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. "Zerocoin: Anonymous Distributed E-Cash from Bitcoin". In: *IEEE Symposium on Security and Privacy*. San Francisco: IEEE, 2013, pp. 397–411.

[22] Malte Möser and Rainer Böhme. "Join Me on a Market for Anonymity". In: *Workshop on the Economics of Information Security (WEIS)*. University of California at Berkeley, 2016.

[23] Malte Möser and Rainer Böhme. "Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees". In: *Financial Cryptography and Data Security, 2nd Workshop on BITCOIN Research*. Ed. by Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff. Vol. 8976. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2015, pp. 19–33.

[24] Malte Möser, Rainer Böhme, and Dominic Breuker. "An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem". In: *APWG eCrime Researchers Summit (ECRIME)*. San Francisco: IEEE, 2013, pp. 1–14.

[25] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: https://bitcoin.org/bitcoin.pdf (visited on 2016-06-26).

[26] Jonas David Nick. "Data-Driven De-Anonymization in Bitcoin". Master Thesis. ETH Zürich, 2015.

[27] Andreas Pfitzmann and Marit Köhntopp. "Anonymity, Unobservability, and Pseudonymity – a Proposal for Terminology". In: *Designing Privacy Enhancing Technologies*. Ed. by Hannes Federrath. Vol. 2009. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2001, pp. 1–9.

[28] Ben Reeves. *Use Apache http client + Multiple API Endpoints*. 2014. URL: https://github.com/blockchain/Sharedcoin/commit/550b39 (visited on 2015-08-11).

[29] Fergal Reid and Martin Harrigan. "An Analysis of Anonymity in the Bitcoin System". In: *Security and Privacy in Social Networks*. Ed. by Y. Altshuler, Y. Elovici, A.B. Cremers, N. Aharony, and A. Pentland. New York: Springer, 2013, pp. 197–223.

[30] Dorit Ron and Adi Shamir. "Quantitative Analysis of the Full Bitcoin Transaction Graph". In: *Financial Cryptography and Data Security*. Ed. by Ahmad-Reza Sadeghi. Vol. 7859. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2013, pp. 6–24.

[31] Meni Rosenfeld. *Overview of Colored Coins*. 2012. URL: https://bitcoil.co.il/BitcoinX.pdf (visited on 2016-08-05).

[32] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. "CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin". In: *ESORICS'14*. Proceedings of the 19th European Symposium on Research in Computer Security. Vol. 8713. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2014, pp. 345–364.

[33] Amitabh Saxena, Janardan Misra, and Aritra Dhar. "Increasing Anonymity in Bitcoin". In: *Financial Cryptography and Data Security, 1st Workshop on BITCOIN Research*. Ed. by Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith. Vol. 8438. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2014, pp. 122–139.

[34] Andrei Serjantov and George Danezis. "Towards an Information Theoretic Metric for Anonymity". In: *Privacy Enhancing Technologies*. Ed. by Roger Dingledine and Paul Syverson. Vol. 2482. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2003, pp. 41–53.

[35] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. *BitIodine: Extracting Intelligence from the Bitcoin Network*. Ed. by Nicolas Christin and Reihaneh Safavi-Naini. Vol. 8437. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 457–468.

[36] Peter Todd. *Stealth Addresses*. 2014. URL: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html (visited on 2017-02-10).

Appendix

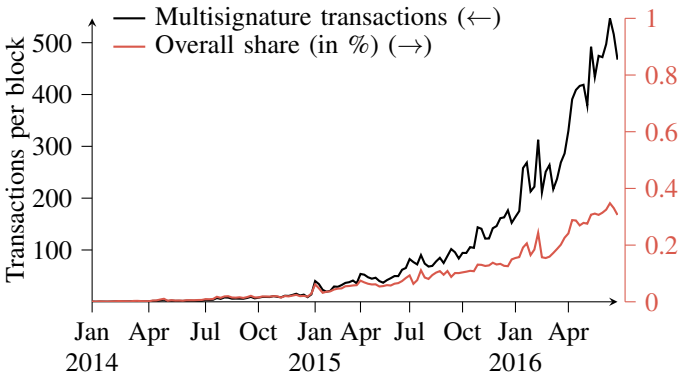


Fig. 10: Number and share of multisignature transactions

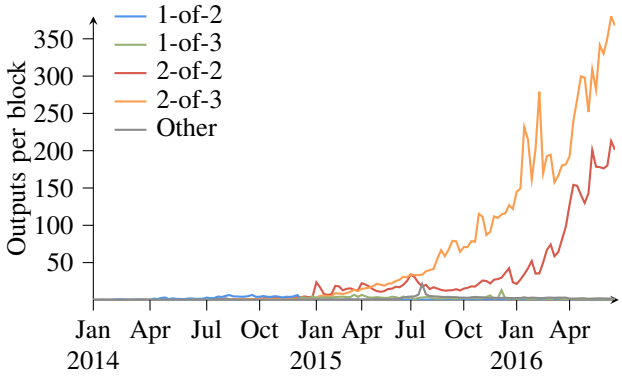


Fig. 11: Number of multisignature-outputs per variant (P2SH multisignature outputs can only be identified after they have been spent)