Forensic Science International: Digital Investigation 52 (2025) 301869

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS EU 2025 - Selected Papers from the 12th Annual Digital Forensics Research Conference Europe

Tumbling down the stairs: Exploiting a tumbler's attempt to hide with ordinary-looking transactions using wallet fingerprinting

Jan Zavřel [*], Michal Koutenský, Daniel Dolejška, Vladimír Veselý

Faculty of Information Technology, Brno University of Technology, Božetěchova 1/2, Brno, 612 00, Czech Republic

A R T I C L E I N F O

Keywords:
Blockchain
Clustering
Cryptocurrency
Mixer
Tumbler
Fingerprinting

A B S T R A C T

The privacy of Bitcoin transactions is a subject of ongoing research from parties interested in enhancing their security, as well as those seeking to analyze the flow of funds happening in the network. Various techniques have been identified to de-obfuscate pseudonymity, e.g., heuristics to cluster addresses and transactions, automatic tracing of transaction chains based on usage patterns/features that may reveal common ownership. These techniques gave rise to services that attempt to make these techniques unreliable with specific forms of behavior. Examples of such behavior include using one-time addresses or transactions with multiple participants. Centralized services employing these behavior patterns, commonly known as *tumblers* or *mixers*, offer customers a way to obfuscate their financial flows. In turn, new approaches have been proposed in recent scientific literature to exploit the way the mixers operate in order to gain insight into the underlying financial flows. In this paper, we analyze some of these approaches and identify challenges in the context of their application to a particular modern mixing service – Anonymixer. Furthermore, based on this analysis, we propose a novel approach for identification of addresses involved in mixing with capability to distinguish between depositing/withdrawing parties and mixer inner addresses. The approach utilizes wallet fingerprints, which we have extracted using statistical measurements of mixer's behavior. An internally developed tool implementing the proposed techniques automates the deobfuscation process and outputs individual money transfers.

1. Introduction

Cryptocurrencies are often perceived as a technology that provides near-instantaneous, unregulated, and cryptographically-secured money transfers between parties. The reliance on cryptography provides anonymity (e.g., Monero, Zcash) or at least pseudonymity (e.g., Bitcoin, Ethereum). As such, cryptocurrencies have pioneered a new way of financial interaction and doing business. Cryptocurrencies have also become the primary tool for a number of illicit activities, including ransomware, procurement of illegal goods, gambling, and sanctions evasion. For these reasons, cryptocurrencies are the subject of research and innovation efforts aimed at improving and reducing their security. Various parties in the ecosystem are competing for and against more law enforcement (e.g., police in need of successful progress in crime investigation (Europol)) and regulatory oversight (e.g., governments using distributed ledger system (Schwalm et al., 2022)). The considerable success of darknet marketplaces, ransomware campaigns, and digital scams/frauds has prompted interest from law enforcement agencies (LEAs), which in turn has led to the stimulation of development in the area of cryptocurrency traceability.

To address the challenges posed by the identification of relations between cryptocurrency addresses and transactions (particularly the impact of co-spent clustering (Meiklejohn et al., 2016; Möser and Narayanan, 2021), a range of counter-techniques have emerged to obfuscate money transfers. The capacity of Bitcoin-like cryptocurrencies to consolidate numerous inputs and outputs in a single transaction has provided a foundation for *CoinJoin* and mixing.

There are two approaches to transaction obfuscation on the Bitcoin base layer in the current landscape: a centralized one (mixers), and a decentralized one (*CoinJoin*). Users of mixers send their coins to an address provided by the service, with a promise of receiving an appropriate amount of coins later at a withdrawal address they provided. Deposit and withdrawal transactions can be causally linked, but modern mixers typically execute user withdrawals using different coins; behavior sometimes referred to as *swapping*. Centralized mixers have a reputation for disappearing, be it by an exit scam or a forceful shutdown by a LEA (as was the case of the infamous *ChipMixer*). The second on-chain obfuscation approach is trustless but may require some extra

* Corresponding author.
E-mail addresses: izavrel@fit.vut.cz (J. Zavřel), koutenmi@fit.vut.cz (M. Koutenský), dolejska@fit.vut.cz (D. Dolejška), veselyv@fit.vut.cz (V. Veselý).

https://doi.org/10.1016/j.fsidi.2025.301869

Available online 24 March 2025

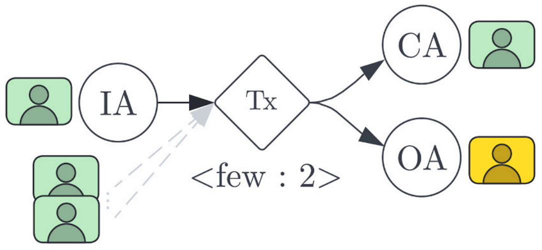

Fig. 1. Transaction <few:2>. The most common interpretation is *simple spend* - a small number of input addresses (*IA*), one change address (*CA*), and one output address (*OA*).

effort from its users – *CoinJoin*. Users use third-party tools to negotiate joined transactions; each user provides a set of inputs and has a claim on a set of outputs. The magic is, or rather should be, that an outside observer cannot link the inputs of a specific user to a set of their outputs with nothing but pure chance (this should be true even for the participants with one another, but it's not trivial to achieve perfectly). The most known example is probably *Wasabi Wallet*, which recently stopped providing this service, as some other providers of the same service face jail. The next best example is thus *JoinMarket*.

The article's primary contribution is the development of a novel approach for identifying addresses involved in cryptocurrency mixing services, particularly in the context of centralized mixing service, *Anonymixer*. This approach employs wallet fingerprinting, a technique that utilizes statistical measurements of mixer behavior, to differentiate between depositing and withdrawing parties, as well as the internal addresses of the mixer. The accompanying tooling facilitates the automatic identification of parties involved in mixing, thereby enhancing the ability to classify and trace financial flows within mixers. This article presents a novel method for enhancing the visibility of mixing operations and provides a foundation for further research into cryptocurrency privacy techniques.

This article is organized as follows. Section 2 outlines related work and the state of the art of the technologies used. To establish a common ground for later inspection of transaction patterns within the *Anonymixer* service, section 3 provides an overview of the most common transaction classification types and their interpretations. Section 4 describes in more detail one particular mixer that serves as a basis for subsequent research, including its modus operandi and potential for reverse-engineering. Section 5 describes the construction of wallet fingerprint and its intricacies: component selection and evaluation. Section 6 focuses on application of the fingerprint on the studied mixer and debates the limitations of this study and approach. Section 7 discusses the achieved state and future work.

2. Related work

In this section, we provide an overview of the current state of the art and place this work in the context of the research of others.

One of the first research articles about mixing or money laundering was published by M. Möser et al., in 2013 (Möser et al., 2013). The authors examined three then-relevant services: *Bitcoin Fog*, *BitLaundry* and *Send Shared* (functionality of *Blockchain.info*). To analyze these services, the researchers ran experiments using the offered services as customers. By subsequent analysis of the transaction graph with the added knowledge of inputs and outputs, researchers were able to conclude that *BitLaundry* provided poor anonymity. All of these services have been shut down since then. It is difficult to evaluate how well the de-mixing techniques presented in the paper apply to more recent mixer implementations, as the landscape has transformed significantly: the volume of daily transactions has increased, users are better educated about best practices for preserving their privacy, and are able to pick from a diverse ecosystem of wallet implementations. T. de Balthasar et al. (de Balthasar et al., 2017) studied three mixing services *Darklaunder*, *Bitlaunder* and *CoinMixer* and found them to offer poor anonymity for their customers, as the authors were able to detect a path between wallets and return addresses. M. Prado-Romero et al. (Prado-Romero et al., 2018) approached the identification of Bitcoin mixing accounts by modeling Bitcoin as a social network and outlining anomaly communities. L. Wu et al. (Wu et al., 2021) categorized mixing services into two categories: *swapping* and *obfuscating*. The authors provided four different examples of these services: *ChipMixer*, *Wasabi Wallet*, *ShapeShift*, and *Bitmix.biz* and conduct analysis to estimate the profitability. J. Pakki et al. (Pakki et al., 2021) analyzed 21 mixing services and explored their varying features, advertised, and actual behaviour. A. Shojaeinasab et al. (Shojaeinasab et al., 2023) conducted research on three mixing services, *MixTum*, *Blender* and *CryptoMixer*. The authors observed *peel chains* used to payout the customers of *MixTum* and *Blender* and devised an algorithm, using address types, for their tracking.

G. Kappos et al. (Kappos et al., 2022) contributed greatly to the research of transaction *peel chains*. To track *peel chain* forward and backward through the transaction graph, the authors clustered addresses, extracted address features, and established a change strategy for each cluster using the output index of the change address.

A different obfuscation method, *Coin Join* transaction, provides a potentially trustless way for users to mix their coins. Möser et al. (Möser and Böhme, 2017) studied one such trustless implementation, *JoinMarket*. The authors could estimate the daily volume and even the total value mixed within 13 months. As recently as 2023, H. Schnoering et al. published a research article (Schnoering and Vazirgiannis, 2023) on detecting *CoinJoin* transactions on the Bitcoin blockchain.

While there has not been any published academic research, to the best of the authors' knowledge, delving into extracting fingerprints (default parameters) of software wallets, the Bitcoin community has done a significant amount of work. Recently, one of the Bitcoin Core contributors, Ishaana Misra, published a well-researched post (Ishaana, 2023) on her blog, demonstrating that wallets (in their default settings) produce consistently constructed transactions. As there is a non-negligible amount of various parameters, one can use these parameters to cluster transactions according to the wallet that likely produced them and associate the clusters with a specific wallet implementation. Work by M. Möser et al. (Möser and Narayanan, 2021) on fingerprints was used as a another reference.

Our work builds on the shoulders of these giants as it explores a method of traversing ambiguous *peel chains* of one mixer service using wallet fingerprint with novel and detailed components and *CoinJoin* detection.

3. Overview of transaction types

In order to understand the proposed method for deobfuscating mixers, it is important to provide a robust theoretical background about different types of cryptocurrency transactions, as well as an assessment of the impact on privacy provided. We used the ideas written by OXT Research (ErgoBTC, 2021) as a base for this section, which provides a classification of Bitcoin-like transactions.

This classification aids in more accurately identifying instances when the ownership changes and when it remains unchanged. Since users have great freedom in the way they create their transactions, the following section can never be exhaustive and applicable to every transaction. However, if users behave rationally with the construction of their transactions (i.e., try to minimize their fees, stick with a particular wallet implementation for a longer period of time), they exhibit consistent patterns.

The critical factor for classification is transaction shape — the number of inputs and outputs. Still, supporting factors like address types, transaction parameters, and clustering heuristics influence classification and, especially, interpretation.

Term “transaction shape” is formulated in format <n:m>, where *n* is the number of input addresses and *m* is the number of output addresses. Keyword *few* refers to less than or equal to 5, while keyword *many* refers to more than 5. This threshold was chosen for this article, and it is based on the current input count distribution within the transactions.

3.1. *Few-to-two transaction*

Transaction shape: typically <1:2>, generally <few:2>

Transactions with a single or few inputs and exactly two outputs (depicted in Fig. 1) are typically interpreted as a *simple spend*. While the number of inputs can vary, as the creator of this transaction may have to include additional inputs to produce sufficient value, the number of outputs stays the same. One output address is under the control of the receiver, and the other is the sender’s change output; such assumption stems from the nature of so called unspent transaction output (UTXO) accounting model of Bitcoin-like cryptocurrencies. If only a single address appears both as the input and the output, it is deemed as change, and the other output belongs to the receiver of the funds — such case is referred to as *transaction with trivial change*.

Transactions with shape <1:2> are the most common type of transaction[1], accounting roughly for 75 % of all transactions with a 30-day moving average as of the time of writing.

3.2. *One-to-one transaction*

Transaction shape: <1:1>

Transactions with one input address and one output address, as depicted in Fig. 2, are referred to as *move* or *sweep*. Under the primary interpretation, a single owner owns both the input and output address. There can be various reasons for this behavior, such as an attempt to obfuscate ownership of funds or transfer to a new address type.

As for the obfuscation aspect, if the amount to be transferred is known prior, a user may use a *simple spend* transaction to generate the exact amount and transfer it to a new address under its control. When the transfer is to be executed, only transaction of type <1:1> is required.

In cases where an available UTXO matches the desired value to be transferred plus transaction fee, <1:1> transaction can be used as a *changeless-spend*.

There are thus two common interpretations: either the funds are moved to an address under the sender’s control or, if there has been a prior *simple spent* transaction or the value is expected or matched by chance, the transaction can transfer funds between different users. The chance that such matching UTXO is available is greater when there are more available UTXOs.

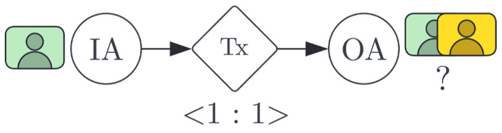

Fig. 2. Transaction <1:1>. The two most common interpretations are *changeless spend* and *move*.

[1] https://transactionfee.info/charts/transactions-1in-2out/.

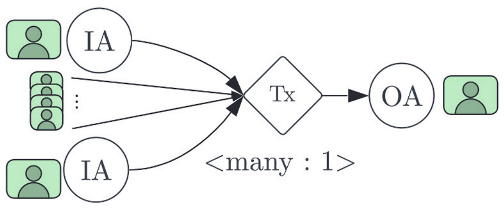

Fig. 3. Transaction <many:1>. The most common interpretation is *consolidation*.

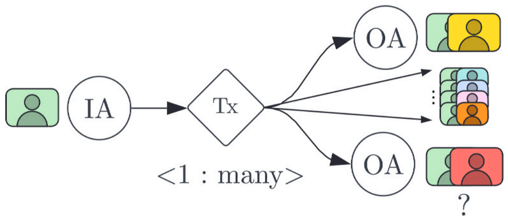

Fig. 4. Transaction <1:many>. The two most common interpretations are *batched spend* and *spread*.

3.3. *Many-to-one transaction*

Transaction shape: <many:1>

As a single Bitcoin owner can have their funds spread across multiple addresses, users may create a transaction that consolidates and centralizes these spread finances to a single address. These transactions typically have a multitude of smaller inputs and a single output, as depicted in Fig. 3. Addresses on both sides of the transaction are deemed to belong to a single user. If, however, the transaction has more than a single output, it can also be interpreted as *consolidation payment*, where the outputs include both the change of the sender and an output for the receiver, or *CoinJoin*, described later in section 3.5, where different owners own their subset of inputs and outputs.

3.4. *One-to-many transaction*

Transaction shape: <1:many>

Conceptually reverse action to a *consolidation* transaction, with a typically single input and a multitude of outputs. It has two different interpretations: *batch spend* or *tokenization/spread*. This type of transaction is depicted in Fig. 4.

In the *batch spend* interpretation, a single user is *batching* multiple payments together. This behavior is typical for services with large flows, such as exchanges, where many users can request withdrawals simultaneously. In such a scenario, output addresses are independent, and there will be a return address among them.

In the *tokenization/spread* interpretation, a single user spreads the funds across multiple addresses under their control to prevent a possible future link between their actions, intending to enhance total privacy and limit traceability.

3.5. *CoinJoin transaction*

CoinJoin transactions are special transactions orchestrated to enhance privacy. While previously mentioned transaction types were always created by a single user (i.e., all addresses on the left side of the transaction can be deemed as belonging to a single user), *CoinJoins* has multiple participants.

There are several popular implementations for *CoinJoin* construction: *JoinMarket*, *Wasabi*, and *Samourai*, each with its own way of constructing *CoinJoin* transactions and differing features. The following subsections outline some of the basic properties of the above-mentioned services. We deem it important to note that Samourai services were seized by LEAs in April of 2024 and Wasabi consequently shut down the operation of their *CoinJoin* coordinator service. However, it is still possible to use *Wasabi* to conduct *CoinJoins* by utilizing an alternative coordinator service. Users may run their own *CoinJoin* coordinator, or use freely available sources[2] to discover other coordinators operated by third parties.

3.5.1. JoinMarket CoinJoin transaction

JoinMarket is a popular wallet implementing *CoinJoins*. A *taker*, party interested in performing a *CoinJoin*, coordinates with a set of *makers*, who provide liquidity for the transactions. The wallet internally uses a concept of *mixdepth*, a wallet structure intended to avoid address reuse.

There are, however, other observations about the generated transactions that can be made. For a *CoinJoin* with N makers, the transactions will: **a)** have $N+1$ equal sized outputs, **b)** have as many change outputs as required, **c)** have at least N inputs. Unlike other wallet providers, *JoinMarket* does not use fixed-size denominations, i.e., the size of transaction's outputs is decided by the taker. The number of makers involved (N) is by default randomly picked between 8 and 10. When it comes to transaction parameters, `nVersion` is set to 2 and `nLockTime` to 0.

3.5.2. Wasabi CoinJoin transaction

Wasabi is another open-source wallet utilizing *CoinJoins* to improve user privacy.

Unlike JoinMarket, Wasabi CoinJoins are restricted to a fixed set of denominations. The list of possible denominations is stated in their website[3]. This makes transactions whose outputs correspond to one of the possible denomination values likely to be Wasabi *CoinJoins*. However, Wasabi can also be used to create *CoinJoins* which do not follow the aforementioned denomination list, although we suspect that such usage is not very common. Wasabi uses 21 as the number of inputs by default and refuses to execute *CoinJoins* with less.

3.5.3. Samourai CoinJoin transaction

Samourai was an open-source wallet with privacy-enhancing features competing with Wasabi. On 25th April 2024, its co-founders have been arrested and the website has been seized by LEAs. The mixing service it provided (named Whirpool) has been shut down.

4. Diving deeper into a modern tumbler

The following section analyses one of the most prominent mixing services with many recommendations. *Anonymixer* announced itself on the BitcoinTalk forum in August 2020, and it is available on the clearweb[4] and darkweb[5]. One of its most stressed aspects is the lack of Cloudflare or similar proxies on the clearweb, which differentiates it from the competition. As these proxies establish a secure TLS connection with the customer instead of the mixer server, data are decrypted on the proxy and typically cached. Proxies thus provide a security concern for privacy-focused users, especially when USA-based proxies, such as Cloudflare, are used.

4.1. Online presence

To gain a better idea about the proclaimed functionality, we visited and read through the official website and profile on *BitcoinTalk* forum as well as *AltCoin Talks* forum.

Online sources mention following key takeaways: **a)** the minimum mixer withdrawal value is 0.0001 BTC; anything below this value is not used to pay out a customer; **b)** The creator is familiar with *CoinJoin*, especially with JoinMarket; **c)** the software wallet is custom-made with obfuscation features; it avoids common clustering heuristics and makes its operation look like regular Bitcoin transactions; **d)** deposited coins are not available for withdrawal (for some other customer) for 18 blocks from the moment of deposit.

4.2. Mixing process

To start a mixing session, customers can enter up to 20 output addresses; *P2TR*[6] addresses are not supported as of September 2024. Next, a mixing session with a lifetime of 72 h is created for the user (identified by a *GUID*). As a privacy-enhancing feature, each entered withdrawal address gets a randomly generated delay before the funds reach the address. The auto-generated delays increase by 0–35 min for each consecutive address, and can also be manually overwritten by the customer if desired. Afterward, the customer is provided a *letter of guarantee*, which contains a summary of the outgoing addresses, their amounts, and delays. Additionally, it includes ten newly created deposit addresses: eight of type *P2WPKH*,[7] one of type *P2SH*[8] and one of type *P2PKH*[9]. The customer is then presented with the total amount (including the fee) to be sent to the mixer. While any of the provided deposit addresses can be chosen for the deposit, the mixer presents *P2WPKH* addresses as the recommended method. After a single confirmation, the service starts sending withdrawal transactions.

To gain insight into the inner workings of the mixer, a total of 22 mixing sessions were carried out: 20 in March 2023 to gain initial data and an additional 2 in September 2024 to verify that the results still held.

The data we collected during these experiments include **a)** address specific attributes – type, number of transactions, delay between trasactions and **b)** transaction specific attributes – number of inputs and outputs, value, signatures, attributes like `nLockTime`, `nVersion`, `nSequence` of inputs and fee per virtual byte (fee/vbyte).

As the often-used words *deposit* and *withdrawal* refer to different actions in each of the following sections, we provide a diagram with explicitly stated names for each case.

We structure the experiments in the following way: firstly, we look at the deposits, withdrawals, and change addresses, respectively. Next, we discuss the observed patterns, and lastly, in the following section 5, we apply the collected knowledge.

4.3. Inspection of deposits

Mixer deposit, in this context, refers to a customer sending coins to address provided by the service. The diagram of the deposit with naming clarification is depicted in Fig. 5.

With our 23 deposits, we were able to uncover that **a)** deposit addresses always has only one deposit and up to one withdrawal transaction, **b)** deposit and withdrawal transactions from the deposit address have delay between 102 and 2231 blocks, averaging 795.26 blocks, **c)** signature sizes vary as one of its components (*r-value)* is not ensured to

[2] For example https://wasabist.io/or https://github.com/Kukks/wasabinostr.

[3] https://docs.wasabiwallet.io/FAQ/FAQ-UseWasabi.html#what-are-the-equal-denominations-created-in-a-coinjoin-round.

[4] https://anonymixer.com/.

[5] http://btcmixer2e3pkn64eb5m65un5nypat4mje27er4ymltzshkmujmxlmyd.onion.

[6] Most recent address type from 2021; address starts with `bc1p`; the output is unlocked by a script and its input (script path spend) or signature by a corresponding private key(s) (key path spend).

[7] Most common type of address; address starts with `bc1q`; output is unlocked with a signature and a public key.

[8] Legacy method of locking output by a customizable script; address starts with `3`; output is unlocked by a script and its input.

[9] Legacy method of locking output to a private-public key pair; address starts with `1`; output is unlocked by a signature and a public key.

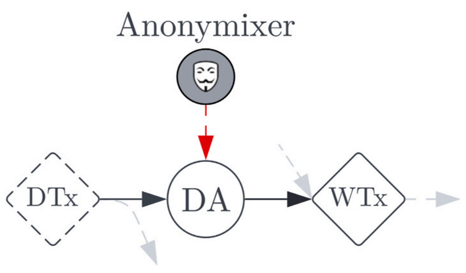

Fig. 5. Diagram of deposit. Deposit transaction (*DTx*), created by user wallet, placed UTXO onto the deposit address (*DA*). Sometime later, the *DA* participates in a withdrawal transaction (*WTx*). Because *DTx* is created by user wallet, it is irrelevant for this inspection.

be as small as possible, **d)** withdrawal transactions were either *JoinMarket CoinJoins* or <1:1> transactions with mostly optimal fees[10], **e)** withdrawal transaction had varying attributes.

4.4. Inspection of withdrawals

Mixer withdrawal, in this context, refers to a customer receiving a payout from the service using a withdrawal transaction. The withdrawal transaction is likely created by the service rather than some other customer, as that would require a steady stream of customers with similar mixing amounts. The diagram of withdrawal, as well as naming clarification, is depicted in Fig. 6.

With our 20 withdrawals, we were able to uncover that **a)** address type of the withdrawal address can be *P2WPKH* (18 cases), *P2SH-P2WPKH*[11] (1 case) or *P2PKH* (1 case), **b)** the number of transactions is equal exactly to two, **c)** the delay between deposit and withdrawal transactions was between 66 and 8979 blocks, averaging 1718 blocks, **d)** signature sizes vary, **e)** transaction fee is mostly optimal, **f)** position of change address between outputs is not stable and **g)** transaction parameters vary for each transaction.

We follow up by inspecting deposits onto the withdrawal address, i. e., transactions that 'prepared' the value onto the withdrawal address for our withdrawal transaction. We concluded that these transactions were, in some cases, created by the mixer and by the customer in others. To validate this point, we provide the follow-up with case studies: two instances where we received the deposit of another user and two different instances where the received value was placed onto the address by the service.

4.4.1. User-created deposit

In two experiments, the value was deposited ('prepared') onto the withdrawal address with <many:many> transactions. We were not able to associate these transactions with any common *CoinJoin* scheme or service. Thanks to available OSINT information[12], we found that inputs into these transactions are associated with *OMG Dark Market*. We thus assume that some deposits into the mixer go directly to someone else as their withdrawals; the mixer can swap coins between users.

4.4.2. Service-created deposit

In two other experiments, the value was deposited ('prepared') onto the withdrawal address by our other withdrawal. In other words, the change address within the withdrawal transaction for one session was later used to execute another, with a different withdrawal for our other session. Thus, we conclude that the mixer can 'prepare' the funds for future withdrawal by itself; the mixer does not only swap coins between users.

4.5. Inspection of change addresses

To complete the initial inspection of the mixer-associated activities, we inspect the change addresses of our withdrawal transaction. The diagram of change address handling, as well as naming clarification, is depicted in Fig. 7.

Out of 20 sessions with 20 withdrawal transactions, 16 contained a change address. While these additional outputs are not guaranteed to belong to the service, we consider it reasonable to assume most of them do, as multiple customers would have to execute withdrawals at the same blockheight with exactly the correct value to produce a changeless transaction of shape <1:2>. In the context of change addresses, we found that **a)** address type was always *P2WPKH*, **b)** it was associated with exactly two transactions — one deposit and one withdrawal, **c)** delay between the two transactions was between 303 and 32 761 blocks with an average of 4 957 blocks, **d)** signature sizes differ, **e)** withdrawal transaction type was either <1:2>, <1:1> or <many:1> and **f)** usually an optimal fee and high variance on all observed transaction parameters.

4.6. Discussion on the observed behavior

Running our experiments, we confirmed that the operator of the *Anonymixer* service provides above-standard service for its users (when compared to previously studied mixers (de Balthasar et al., 2017; Sho-

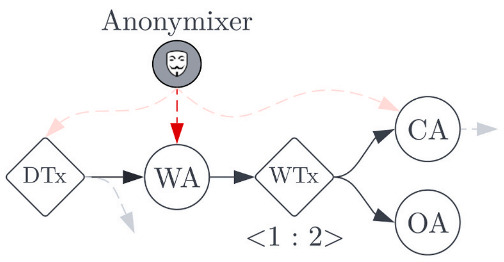

Fig. 6. Diagram of withdrawal. Deposit transaction (*DTx*) placed UTXO onto the withdrawal address (*WA*). Sometime later, the withdrawal address creates a withdrawal transaction (*WTx*), putting UTXO onto the output address (*OA*) and receiving change UTXO on the change address (*CA*). Some withdraw transactions were of shape <1:1> and thus did not include any change address. *OA* is under our control and *WA* under Anonymixers control, owner of *CA* and creator of *DTx* are currently uknown.

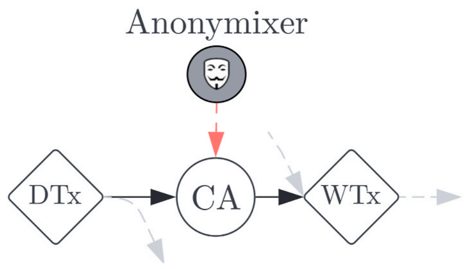

Fig. 7. Diagram of change address handling. Deposit transaction (*DTx*), transaction executing the withdrawal from the service, placed UTXO onto the change address (*CA*). The change address later participates in a withdrawal transaction (*WTx*). These withdrawal transaction were either of shape <1:1>, <many:1> or <few:2>.

[10] Calculated from the average `fee/vbyte` in the transaction's block. We consider optimal fees to be below double the average of the block.

[11] Address type bridging legacy and SegWit addresses; address starts with `3`; output is unlocked by a signature and a public key.

[12] https://tokenscope.com/.

jaeinasab et al., 2023). The custom implementation employs seemingly randomized transaction parameters, making it seem like two subsequent transactions were made by a different user.

In scenarios where user deposits go directly to another user without the involvement of any other address, the visibility of the transaction is limited. The same goes for deposits directly deposited into the *JoinMarket* ecosystem.

When it comes to payouts from the mixer, we observed two different transaction shapes: a) withdrawal using transaction with shape <1:1>; and b) withdrawal using transaction with shape <1:2>. In either case, the value is placed onto the withdrawal address in advance and either directly by another service customer (as their input into the service) or by the service itself (as a change from the previous withdrawal). In cases when the service places a remainder from user withdrawal to a new address, which is later used for another withdrawal, a *peel chain* is created. *Peel chain* refers to a pattern commonly found on the Bitcoin-like blockchains. It refers to a chain of variable length of typically <1:2> transactions, where a larger amount is gradually, through the individual transactions, being 'peeled' and typically sent to various addresses. A general depiction of *peel chaining* is provided in Fig. 8). These chains can be traversed forward and backward once a single participating address is discovered.

In experiments where change address was encountered, it was later withdrawn from using <1:2> — creating another link the in the *peel chain*, <1:1> — end of the *peel chain*, or <many:1> — *reset* of the *peel chain.*

Interestingly, the transfer out from the change address came in three distinct varieties: <many:1>, <1:1>, or <few:2>, with a total of 12 unique transactions observed. Out of the eight consolidation transactions (<many:1>), only four were unique, meaning multiple change addresses from different sessions were later used in a transaction together. This hints that these addresses are under the control of the mixer service, as the mixes were done on different days at different times — it is unlikely that a single customer of the service was mixing at the same exact time on different days on four occasions. Since we are confident that the service performed these *consolidation* transactions, we can later inspect other inputs and track them backward in the transaction history.

The service claims that withdrawals can be batched together if scheduled for the same time. However, we could not verify whether this behavior occurs across different mixing sessions, i.e., for different customers. On the other hand, we later confirmed that the behavior occurs within the same mixing session by specifying withdrawals to occur simultaneously (by setting the same delay for two payout addresses). As a result, the withdrawal transaction has three outputs: two of our addresses and one change address.

Overall, *Anonymixer* provides a unique case for mixers, as there are some important differences when compared with previously popular mixing services, such as *ChipMixer* or other researched mixers with poor implementation, such as *MixTum.*

Chipmixer, for example, did not attempt to hide its operation in a sufficient manner. Inputs were typically consolidated and tokenized in one step, using a <many:many> transaction. The resulting tokens were then distributed to the customers for their withdrawal; hard to trace but easy to detect.

A different mixing service, *MixTum*, used consolidation transactions

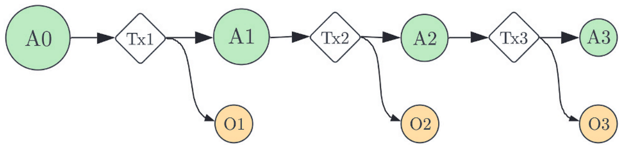

Fig. 8. Generic depiction of a peel chain. The initial value from address A0 is being "peeled" by individual transactions with subsequently smaller output values, sequential creating additional outputs O1, O2 and O3.

to gather inputs (<many:1>). Afterward, customer withdrawals were created using a long peel chain pattern that reused the same address type, allowing for relatively easy tracking.

Anonymixer, on the other hand, uses more complicated mixing schemes. Firstly, inputs into the service are sometimes mixed with independent users' coins, making tracking any withdrawal transaction difficult, as the pool of addresses is "diluted" by random users. Secondly, transactions are created with a variety of parameter and feature combinations. Thirdly, withdrawal transactions may directly follow a deposit from a customer, disallowing peel chain tracking backward through transaction history. Moreover, the *peel chain* cannot be traversed forward if the withdrawal transaction only has a single output.

If we are to identify inputs and outputs into the *Anonymixer*, we need a reliable method for traversal of the *peel chains* it creates. The success of chain traversal is highly dependent on correctly identifying the change address in each link.

While some well-known methods can work on *Anonymixer* in specific cases, the encountered chains are often too ambiguous — fresh addresses on the outputs, same address types, roughly equal and rounded output values, and similar future spending transactions. To gain more vectors for change address identification and untangling of the *peel chains*, we create a *wallet fingerprint* of the *Anonymixer* wallet, as it is — by the author's own testimony — created "from scratch".

5. Wallet fingerprinting

Wallet fingerprinting aims to identify repeated and stable parameters within transactions that a particular wallet generates and broadcasts to the network. Fingerprinting generally applies only when the default wallet settings are used, as many wallets expose these settings as configurable to users, even though some wallet fingerprints stem from limitations in the implementation and features. Creating a trustworthy and detailed fingerprint allows us to track peel chains even on transactions where other typical techniques, like address type matching, value rounding, and clustering, would not be as reliable.

Fingerprints are constructed from a set of components and their expected values. As the service uses its own wallet implementation, we could not apply any known fingerprints associated with commonly used wallets (Ishaana, 2023) (e.g., Bitcoin Core, Electrum), but we had to create our own based on the previously described experiments. Since we found that the components commonly used to create fingerprints are insufficient for this use case, we expanded them in certain directions.

The following section describes how we approached our components' selection and often dives into technical details.

5.1. Component selection

Based on our findings, the mixer wallet seems to randomize several parameters, such as `nVersion`, `RBF`, the value of `nLockTime`, and whether the `nLockTime` value is applied or not. Other wallets are not known to behave in such a manner; transactions generated by a single wallet are relatively stable (even though an article (Consent, 2022) states that the Bitcoin Core wallet creates transactions with different parameters when RPC calls are used).

The selected components for the fingerprint are shown in Table 1. We considered the following transactions for the construction of the fingerprint that we collected in the previous section:

a) The withdrawal transaction from the deposit address (non-CoinJoin);
b) The withdrawal transaction towards our withdrawal address; and
c) The withdrawal transaction from the change address.

Withdrawal transactions from the change addresses admittedly have a lower amount of confidence. However, none exhibited any deviation from the already known fingerprints regarding transaction parameters.

Table 1
Wallet fingerprint components, their observed values and quantities. Numerous parameters are omitted. In terms of input/output orderings, we recognize value, lexical and historical.

Abridged Wallet Fingerprint	
General Transaction Parameters	
nVersion	{1, 2}
Tx types	simple spend, move consolidation
RBF	{true, false}
Low-r-Grinding	×
Input/Output Attributes	
Particular Input Ordering	×
Particular Output Ordering	×
Deterministic Change Index	×
Change Address Type Match	×
Handling of nLockTime	
nSeq. 0xFFFFFFFF	6x *none*, 2x *height*
nSeq. 0xFFFFFFFE	7x *none*, 10x *height*
nSeq. 0xFFFFFFFD	6x *none*, 9x *height*
nSeq. other	0x *none*, 0x *height*
Unix time value	×
Address Attributes	
Number of transactions	2
Blocks between transactions	66 ≤
Address Types	P2WPKH, P2PKH P2SH-P2WPKH

Since these were the only non-CoinJoin transactions with multiple inputs that we assume the mixer created, it allowed us to inspect input ordering.

5.2. Discussion on the extracted fingerprint

To get an idea of what transaction parameters and what combinations are common, we scanned the last 1000 blocks around the time of mixing in March of 2023. The results are shown in a graphical form in Fig. 9 and with concrete values in Table 2.

At first glance, the service operator has certainly gone to some

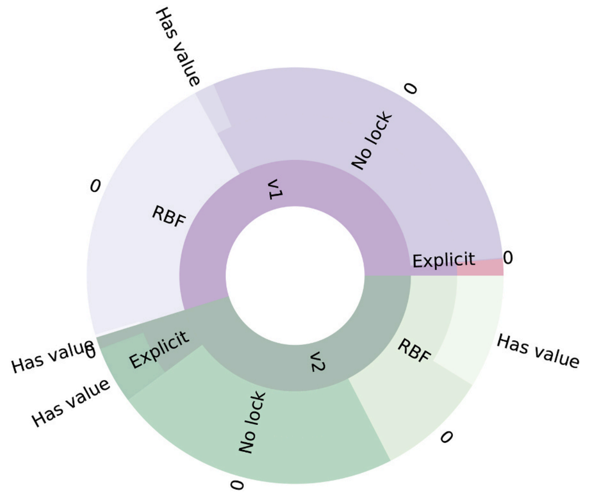

Fig. 9. Visual representation of transaction distribution based on height-lock field values. In this figure, you can see the distribution of values in various height-lock related fields. The innermost ring splits the transactions by value of the `nVersion` field (1 or 2). The middle ring further splits the segments by the `nSequence` field values (where "No lock" corresponds to `nSeq.==0xFF..FF`; "Explicit" to `nSeq.==0xFF..FE` and "RBF" to `nSeq.<=0xFF..FD`). The outermost ring splits the segments by values of `nLockTime` itself. Concrete counts and percentages can be seen in Table 2.

Table 2
Distribution of latest transactions based on their values of `nVersion`, `nSequence` and `nLockTime`. The statistics were calculated based on all transactions from blocks (mined over roughly one week) `799,000–800,008`. Abnormal combinations of parameters are highlighted.

nVer.	nSeq.	nLockTime	Count	Percent
v1	No lock	0	651 143	30.034618 %
v2	No lock	0	485 413	22.390157 %
v1	RBF	0	468 101	21.591624 %
v2	RBF	Has value	190 996	8.809880 %
v2	RBF	0	186 733	8.613245 %
v2	Explicit	Has value	96 894	4.469332 %
v1	**No lock**	**Has value**	**33 890**	**1.563210 %**
v1	**Explicit**	**0**	**28 056**	**1.294111 %**
v2	**Explicit**	**0**	**19 757**	**0.911311 %**
v1	RBF	Has value	5 141	0.237134 %
v1	Explicit	Has value	1 802	0.083119 %
v2	**No lock**	**Has value**	**49**	**0.002260 %**

lengths to provide an above-standard level of privacy. Typical fingerprinting candidates, such as `nVersion` and `RBF`, are not very useful in isolation, as their value is set most likely randomly, blending in with the rest of the transactions on the network. However, certain combinations with other parameters are fairly uncommon, according to Table 2.

The parameter `nVersion` of 2 is common by itself. However, when combined with explicitly disabled *locktime* (`nSequence 0xFF..FF`) with `nLocktime` value of blockheight, the likelihood of encountering such a transaction is very low. Yet, we detected this combination of parameters. The same goes for cases where `nVersion` is 2, *locktime* is explicitly enabled (`nSequence 0xFF..FE`) but `nLocktime` has a value of 0. We observed this combination multiple times.

We therefore assume that the *Anonymixer* wallet generates the values of transaction field `nVersion`, `nSequence` of inputs and `nLockTime` seemingly independently or in wrong-at-the-time ratios, which results in rare combinations that do not commonly appear in other transactions. For example, `nLockTime` is set on the transactions to a blockheight above 0, but disregarded by all its inputs, or vice-versa: *Locktime* is enabled by inputs and set to 0 on the transaction level. The application of the fingerprint is still *probabilistic* — a transaction matching the fingerprint does not automatically belong to the mixer. Tables 1 and 2 convey the important observation: *Anonymixer* uses fairly uncommon parameter combinations too frequently, allowing us to better identify when the transaction is still likely created by the mixer and when it's likely to be a different user. We can further improve the likelihood of correct identification by utilizing additional context. When examining a link in a standard *peel chain*, we can try to apply the fingerprint in all directions — withdrawal transaction on outputs and deposit transaction of the input — giving us a much better chance, when combined with other anomalies, to correctly differentiate between user and mixer addresses.

Our fingerprint relies on data from addresses as well as transactions. The total address search space is limited, since it focuses only on addresses with two transactions and only three transaction shapes. To limit the number of matched addresses when consolidation is encountered, we utilize no *low-r-grinding* and no input orderings for the classification.

The absence of *low-r-grinding* can also be used to classify transactions with a single input. However, unless adjusted for probability, the result can have a lot of false positives.

While we suspect that the service mostly uses P2WPKH addresses, we cannot completely disregard addresses of other types as we assume they could be used as another type of obfuscation of the *peel chain*.

5.3. Leveraging fingerprint to interpret peel chains

To test the capabilities of our fingerprinting approach, the following section demonstrates its ability to navigate through a transaction peel chain encountered in one of our experiments. This partial transaction graph consists only of addresses that, unless stated otherwise, participated in only two transactions in total, are of the same type, and their transactions split input values into roughly equal parts. Fig. 10 depicts the transaction diagram for this particular case study.

We start on our address `A0`, where we received the withdrawal funds. Since the withdrawal transaction is of shape <1:1> and the input address `A1` into this transaction participated in only two transactions, we follow up to inspect this address. The first question is whether address `A1` could be a deposit address for some customers. To answer this question, we move to inspect the transactions that deposited funds onto it, transaction `Tx1`. Transaction `Tx1` is of shape <1:2> and, importantly, a) `nVersion` is `1` b) `nSequence` of input is *final* but `nLockTime` value is set. This leads us to believe the mixer constructed the transaction, as the combination is invalid. If we inspect the other output address `A2`, we find it to have participated in multiple transactions. Transactions spending UTXO associated with the address, i.e., constituted by the owner of the address, have unsurprisingly extremely stable parameters. From its transactions, we also find that it participates in *dusting attacks*[13], leading us to believe it to be an output address of a different customer and thus address funding the withdrawal transaction, address `A3` belongs to the mixer. If we follow the transaction that deposited funds onto `A3`, transaction `Tx2`, we find the exact same parameters as with `Tx1`; now, however, with signature with high `r-value`. If we follow the other output to address `A4` and inspect its output transaction, we find more standard parameters - as the `nLockTime` value is not set and not enabled. The address `A4` participated in only one other transaction `Tx3`, sending the received UTXO to *Wasabi Wallet* address `A5`, as we detect `Wasabi` CoinJoin by its special characteristics. We thus deem address `A4` as another mixer output, `Tx2` as being created by the mixer, and `A6` as being under the control of the mixer. If we follow our search towards the overall source of the funds, to transaction `Tx4`, which parameters are different but still consistent with mixers wallet fingerprint and follow its other output address `A7` and we inspect its only other transaction `Tx5`, we can identify transfer to the *coinsbuy* platform[14] using address `A8` and to a different address `A9`, whose fingerprint is consistent with address `A7`. We thus deem `A7` as another customer of the mixer service and a customer of the *coinsbuy* platform, `Tx4` as being constructed by the mixer, and `A10` as being under the control of the mixer aswell. We then move back to the main *peel chain* and follow the transaction `Tx6`, to find the first really strong deviation from our collected fingerprint – two inputs from the same address with otherwise standard transaction parameters `nVersion` 2, enabled LockTime with correct blockheight value, leading us to mark this transaction as not generated by the service but by a customer. The exploration is thus concluded as there are no more candidate transactions or addresses to consider.

As we applied the above-described method to all of our captured sessions, we identified a *spread* transaction (<1:many>) as a source of certain chains. After inspection of the only input address, we found an exit from *JoinMarket*, which we know is used by *Anonymixer*. When we applied our fingerprint to other outputs from the *spread* transaction, we identified multiple other origins of *peel chains*, all of which matched our fingerprint exactly — many instances of specific rare combinations of `nVersion`, `nSequence` and `nLocktime`.

As a result, we consider adding *spread* into the fingerprint as an improvement for future experiments and evaluations.

6. Automation of the mixer address discovery process

As demonstrated in the previous section, using the collected wallet fingerprint allows us to track seemingly perfect *peel chains* — chains whose addresses are of the same type with no address reuse and whose transaction split the input value into roughly equal parts. This makes the detection of change address using standard approaches difficult. The last step in showcasing the functionality of our proposed approach is to automate the process. The goal is to:

a) identify addresses of the service;
b) identify withdrawal addresses of customers of the service;
c) identify deposit transactions into the mixer;

To gather blockchain data, we utilize a *Bitcoin Core*[15] client and the *Mempool*[16] service, as these provide transaction links forward and backward between inputs and outputs.

We use the addresses collected during the experiments as a seed for our algorithm. Essentially, we perform forward and backward searches from the seeds. We subject each encountered address and transaction to the fingerprint, extracted in Section 5, and either continue the search or mark the transaction or address as an *anomaly*, possible withdrawal or deposit, depending on the context.

This search is enabled by the unique characteristics of the mixer itself. Since it is trying to hide its operation under the guise of standard Bitcoin transactions — that have only a small number of inputs and outputs — brute-force search can be considered a viable strategy.

6.1. Results

By traversing the transaction graph backward from the withdrawal addresses, forward from the deposit addresses, and forward from the change addresses, we were able to identify 344 service addresses and 461 customer addresses. The total volume of Bitcoin transferred through the encountered *peel chain* was `0,8534 BTC`, the earliest encountered transaction was 17th January 2023, the latest 15th May 2024.

While we know that the estimated volume is far below the actual value, it acts as the lower bound. As we were able to identify the addresses of hundreds of supposed customers, we also confirmed, using OSINT[17] methods, that on several occasions, the customer withdrawal money could be followed directly into well-known online services, such as *HTX*, *coinsbuy* or *ChangeNOW*. In several other cases, the output addresses we associated with reports on well-known community sites[18] typically with scams or ransomware.

With this brief example, we aim to emphasize that we can traverse peel chains both forward and backward through the transaction graph, even though the visibility is still quite limited since ambiguous cases are ignored. When <1:2> transaction is encountered from either direction, one has to correctly determine which output is the change address and which output is the *peeled* value. Wallet fingerprints can help us in cases where clustering heuristics or commonly used change address detection heuristics do not work, especially when the wallet uses such specific combinations as is our case.

Because the studied peel chain often ended or started with *consolidation* instead of <1:1> transaction, we were able to uncover tens of addresses from a single seed.

Like numerous other studies, the absence of extensive testing/validation sets acts as a barrier, preventing us from accurately assessing this method's effectiveness. However, given that our assumptions about the service are conservative, we can estimate the lower bounds.

One of the obvious limitations of our approach of analysis of *Anonymixer* is the inability to detect isolated transfers: if a value is deposited into the mixer and then immediately, without intermediate transaction,

[13] Distribution of tiny amount of currency for tracking purposes. Wallets often automatically involve available UTXOs in future transactions, exposing its owner to the co-spent heuristic.
[14] https://coinsbuy.com/.
[15] https://bitcoincore.org/.
[16] https://mempool.space/.
[17] https://tokenscope.com/.
[18] http://checkbitcoinaddress.com/.

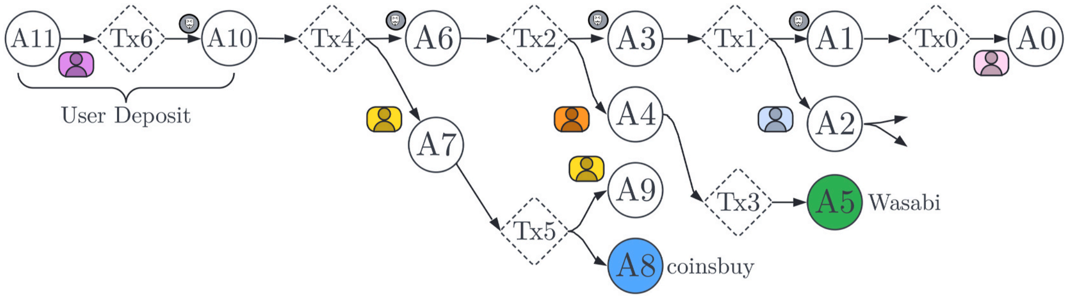

Fig. 10. Transaction graph depicting the studied *peel chain*. Using the constructed fingerprint, we have a new vector to distinguish between different actors.

used in its whole for withdrawal for some other customer (<1:1> *changeless* payment), no input or change address can be followed. In some such specific cases that we got the chance to observe, the mixer had no financial profit — change address provides a way for the mixer to take its cut, either as profit or as a portion of some customer's payout — the traceability is limited, and so is the mixer's profitability.

We believe that we encountered this exact scenario on several occasions — our deposited value enlarged by the mixers fee was deposited onto the mixers address and later, using *changeless spend*, used to payout another customer with the same value we originally intended to mix. The whole service fee was thus spent on the transaction fee.

This holds when the number of mixed coins is on the lower side of the spectrum - as the service fee is calculated from the total amount being mixed while the transaction fee is relatively constant.

To provide a performance comparison with existing approaches, all the techniques discussed in Section 2 would need to be implemented and tested on the same dataset. However, since our method — analysis of uncommon parameters and subsequent construction of fingerprint — can work in tandem with the current methods, the performance of *peel chain* tracking heuristic should only increase — as we simply provide additional vectors to feed the heuristic to influence decisions.

7. Conclusion

Usage of the *Anonymixer* unique wallet fingerprint has proven to be a valid approach, as we exploited its high variance in transaction parameters and automatized identification of involved addresses in mixing rounds and peel chaining. Hence, wallet fingerprinting holds significant potential for classification purposes, despite the limited exploration of this concept in contemporary scientific literature. This approach appears promising in several ways. For example, it could be employed: a) to verify that a common-spent clusters do not contain irrelevant addresses; b) to improve detection of spare-changes address within transactions; or c) to provide an additional vector for correlation of blockchain transactions with off-the-blockchain events.

Nevertheless, the generation of a fingerprint from a limited set of approximately 40 transactions, with the objective of monitoring a specific service on the blockchain, represents a significant advance. The techniques and approaches that we have presented can, and it is highly probable that they will, be employed in the future to enhance the efficacy of change address detection heuristics, thereby improving visibility at the base layer of the Bitcoin network.

Raw data from mixing sessions are available to cybersecurity researchers upon request and validation of serious and honest interest.

Acknowledgement

This article was supported by the project *MIXER: Building community interested in cryptocurrency mixers*, code VJ03030037, funded by Ministry of the Interior of the Czech Republic.

References

Consent, 2022. Consent Wallet Fingerprinting Using nLocktime and nVersion [online] Available at: https://consentonchain.github.io/blog/posts/fingerprinting/. (Accessed 15 December 2024).
de Balthasar, T., Hernandez-Castro, J., 2017. An analysis of Bitcoin laundry services. In: Lipmaa, H., Mitrokotsa, A., Matulevičius, R. (Eds.), Secure IT Systems. Springer International Publishing, pp. 297–312.
ErgoBTC, 2021. Understanding Bitcoin Privacy with OXT — Part 1. OXT Research [online], Available at: http://web.archive.org/web/20240420043914/. (Accessed 15 December 2024) https://oxtresearch.com/understanding-bitcoin-privacy-with-oxt-part-1/.
Europol. (n.d.). One of the darkweb's largest cryptocurrency laundromats washed out. [online] Available at: https://www.europol.europa.eu/media-press/newsroom/news/one-of-darkwebs-largest-cryptocurrency-laundromats-washed-out. [Accessed 15 December. 2024].
Ishaana, 2023. Wallet Fingerprints: Detection & Analysis [online] Available at: https://ishaana.com/blog/wallet_fingerprinting/. (Accessed 15 December 2024).
Kappos, G., et al., 2022. How to peel a million: validating and expanding Bitcoin clusters. In: 31st USENIX Security Symposium (USENIX Security 22). USENIX Association, Boston, MA, pp. 2207–2223. Available at: https://www.usenix.org/conference/usenixsecurity22/presentation/kappos.
Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S., 2016. A fistful of Bitcoins. Commun. ACM 59 (4), 86–93. https://doi.org/10.1145/2896384.
Möser, M., Böhme, R., 2017. The price of anonymity: empirical evidence from a market for Bitcoin anonymization. J. Cybersecur. 3 (2), 127–135. https://doi.org/10.1093/cybsec/tyx007.
Möser, M., Narayanan, A., 2021. Resurrecting address clustering in Bitcoin. CoRR abs/2107.05749. Available at: https://arxiv.org/abs/2107.05749.
Möser, M., Böhme, R., Breuker, D., 2013. An inquiry into money laundering tools in the Bitcoin ecosystem. In: 2013 APWG eCrime Researchers Summit, pp. 1–14. https://doi.org/10.1109/eCRS.2013.6805780.
Pakki, J., Shoshitaishvili, Y., Wang, R., Bao, T., Doupé, A., 2021. Everything you ever wanted to know about Bitcoin mixers (but were afraid to ask). In: Borisov, N., Diaz, C. (Eds.), Financial Cryptography and Data Security. FC 2021. Lecture Notes in Computer Science( ), vol. 12674. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-64322-8_6.
Prado-Romero, M.A., Doerr, C., Gago-Alonso, A., 2018. Discovering Bitcoin mixing using anomaly detection. In: Mendoza, M., Velastín, S. (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2017, Lecture Notes in Computer Science( ), vol. 10657. Springer, Cham. https://doi.org/10.1007/978-3-319-75193-1_64.
Schnoering, H., Vazirgiannis, M., 2023. Heuristics for Detecting CoinJoin Transactions on the Bitcoin Blockchain. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2311.12491.
Schwalm, Steffen, Albrecht, Daria, Alamillo, Ignacio, 2022. eIDAS 2.0: Challenges, Perspectives and Proposals to Avoid Contradictions between eIDAS 2.0 and SSI. Open Identity Summit 2022. Gesellschaft für Informatik e.V., Bonn, pp. 63–74. https://doi.org/10.18420/OID2022_05. PISSN: 1617-5468. ISBN: 978-3-88579-719-7, Regular Research Papers. Copenhagen, Denmark. 07.-08. July 2022.
Shojaeinasab, Ardeshir, Motamed, Amir Pasha, Bahrak, Behnam, 2023. Mixing detection on Bitcoin transactions using statistical patterns. IET blockchain 3 (3), 136–148. https://doi.org/10.1049/blc2.12036.
Wu, L., et al., 2021. Towards understanding and demystifying Bitcoin mixing services. In: Proceedings of the Web Conference 2021. Association for Computing Machinery (WWW '21), New York, NY, USA, pp. 33–44. https://doi.org/10.1145/3442381.3449880.