CoinJoin ecosystem insights for Wasabi 1.x, Wasabi 2.x and Whirlpool coordinator-based privacy mixers

Petr Svenda
Masaryk University
svenda@fi.muni.cz

Jiri Gavenda
Masaryk University

Vasilios Mavroudis
The Alan Turing Institute

Chris Hicks
The Alan Turing Institute

Abstract

Coinjoin is a collaborative Bitcoin transaction designed to enhance users' privacy by joining coins of multiple parties. Coinjoin implementations with a centralized trust-minimized coordinator have mixed more than 391,000 bitcoins (₿) since 2018. In June 2024, law enforcement actions led to the shutdown of coordinators of all three leading coinjoin designs—Wasabi 1.x, Wasabi 2.x, and Whirlpool—prompting a proliferation of new independent coordinators.

Prior studies of this ecosystem offer little information beyond coinjoin detection and aggregation of related statistics, primarily due to the unavailability of information (intentionally) obscured by coinjoins. To overcome this limitation, we combine on-chain transaction analysis, coordinator monitoring, and active coinjoin participation, complemented by tailored analysis and visualizations, to: 1) provide a model for estimation of the number of active users at a time, 2) detect previously unknown coordinators, 3) retrospectively observe activity patterns of prominent parties, and 4) deliver continuously updated insights into the ecosystem.

We show that while Wasabi 1.x remains inactive and recently resumed Whirlpool exhibits little activity, the Wasabi 2.x mixing has exceeded pre-shutdown levels under the new coordinator *kruw.io*, which now accounts for 99% of mixed inputs (over 46,000 ₿ in the past 18 months) with raising concurrently active users now estimated to 70+ in average coinjoin, providing the first data-based estimate of the participants' anonymity set. We revealed a previously unknown but significant coordinator active from early May to November 2024, with a method capable of identifying emerging coordinators. Although WW2 conceals large-input wallet activity more effectively than WW1, it is still detected by the proposed transaction-centric liquidity analysis.

Keywords

CoinJoin, Bitcoin, Privacy, Wasabi Wallet, Whirlpool

1 Introduction

Coinjoins serve to increase the anonymity of financial assets within the UTXO-based cryptocurrencies [15]. They help to protect participants against large-scale user monitoring, financial history profiling from target-seeking criminals, and discriminatory access to financial services. At the same time, they have also been used for obscuring movement of stolen or illegally obtained digital assets, notably by North Korea's Lazarus group on several occasions, including the recent February 2025 $1.5 billion Bybit hack [5]. From 2018 until June 2024, Wasabi 1.x (WW1), Wasabi 2.x (WW2), and Samourai Whirlpool (SW) represented the leading coinjoin implementations relying on a centralized coordinator. SW was abruptly shut down by the FBI in April 2024, followed by a voluntary shutdown of the zkSNACKs' WW2 coordinator on June 1. The zkSNACKs coordinator was subsequently replaced by new, smaller coordinators. These events provide a timely opportunity to evaluate the operational history and privacy guarantees of the coinjoin designs.

In general, packet mixing protocols [6] aim to obscure the linkage between output and input messages, protecting participants from attackers who observe or interfere with the protocol's execution. Privacy depends on the anonymity set (i.e., the plausible owners/recipients of an output) typically correlated to the number of active users. Coinjoin protocols apply this idea to on-chain transactions, where a group of users creates a single collaborative transaction, hindering the transaction's input-output linkage by attackers. So far, no data-based estimation of the number of active users in the coinjoin designs was provided in open literature.

Initial studies in this field focus on the on-chain detection of coinjoin transactions and gathered statistics on their prevalence (e.g., [7, 14, 17, 18, 20, 23]). Failure to recognize that a coinjoin involves inputs from multiple distinct wallets can lead clustering algorithms [13] to erroneously merge unrelated address groups, thus compromising the resulting data's accuracy and usability. Because coinjoins are used by both lawful and unlawful actors, a more accurate estimation and interpretation of anonymity from on-chain data are necessary to avoid a one-sided interpretation.

Stütz et al. [18] analyzed WW1 and SW activity up to early 2022, while Wu et al. [26] modeled the economic incentives of coordinator-based mixing. The Dumplings project [7] implemented the extraction of coinjoin transactions based on various heuristic approaches from published works. Previous studies assumed only well-defined and public coordinators; that situation changed in May 2024, after which monitoring services such as LiquiSabi [21] and Wabisator [1] emerged to inform users about the activity of known post-zkSNACKs WW2 coordinators; with their scope limited only to real-time summaries. No method for detecting unknown, non-public coordinators has been devised so far.

Proceedings on Privacy Enhancing Technologies 2026(2), 557–592

https://doi.org/10.56553/popets-2026-0061

To overcome the aforementioned gaps, we investigate the following research questions:

RQ1: Which previously unknown or non-public coinjoin coordinator services can be retrospectively identified using on-chain observable transaction behavior?

RQ2: What temporal characteristics of coinjoin usage can be estimated from on-chain data?

RQ3: How can the popularity, liquidity, and transaction frequency of major coinjoin designs be retrospectively analyzed?

Contributions. To answer our research questions, we make the following contributions:

- We present a practical method for identifying, filtering, and pre-processing on-chain coinjoin transactions that, when combined with the new coordinator attribution algorithm, enables both retrospective and prospective detection of previously unknown, non-public coordinators (Sect. 4).
- Using real input and output distributions gathered from long-term client-side participation experiments (Sect. 3), we develop a data-driven estimator for the number of active wallets, achieving practical accuracy with tight error bounds (Sect. 4).
- We provide detailed insights into the operation and evolution of three major coordinator-based coinjoin designs and their coordinators, highlighting patterns of continuity following the FBI-led shutdowns in 2024 (Sect. 5).

We release all processing tools[1] as well as a continuously updated dataset[2] about past and currently active coordinators to support informed decisions of privacy-seeking users.

Table 1: Basic definitions

Term	Explanation
coin	Transaction output (TXO), whether spent or unspent (UTXO), for which a wallet controls the spending key; term used for readability.
anonset	Number of same-denomination outputs from WW1 transaction, i.e. $anonset(o) = \lvert\{o_i \in O : o_i = o\}\rvert$, where O is the set of all coinjoin outputs and = tests equality of outputs' values; the anonset is computed locally by each WW1 wallet for every UTXO (default target: 50).
anonscore	A more advanced anonymity metric computed locally by WW2 wallets for each UTXO. It is based on the number of equal-value outputs, but also considers the anonscore of inputs and the possibility that one user owns multiple same-valued outputs. (default target: 5; raised to 10 in v2.5.1, Jan. 2025). Let the value-weighted average anonscore of inputs of a user u be denoted as a_u, the minimum anonscore of his inputs as m_u, and let O_u be a set of his outputs, and O be the set of all outputs. Then the anonscore of $o \in O_u$ is computed as $as(o) = \begin{cases} m_u, & \text{if } \lvert\{o_i \in O : o_i = o\}\rvert = 1 \\ a_u + \frac{\lvert\{o_i \in O : o_i = o\}\rvert}{\lvert\{o_i \in O_u : o_i = o\}\rvert}, & \text{if } \lvert\{o_i \in O : o_i = o\}\rvert > 1 \end{cases}$
privacy progress	Value-weighted ratio of the total *anonscores* of all UTXOs in a WW2 wallet to a target *anonscore*. A value of 100% indicates that all UTXOs have reached or exceeded the specified target.
remix rate	Proportion of coinjoin inputs that are remixed (outputs of another coinjoin transaction) relative to all inputs, measured either by input count or total input value (in ₿).
burn time	Absolute (wall-clock) or logical (number of intermediate coinjoins) time elapsed between output creation and its subsequent use as an input in a later coinjoin transaction; captures age of the inputs used.
standard denomination	Shared list of concrete values (in ₿) used by all other participating wallets for coinjoin transaction outputs to increase the likelihood of a group of outputs with exactly the same value (i.e., denomination).
large wallet	A wallet that registers and remixes inputs of a significantly higher value (multiples) than from all other active wallets combined (for the coinjoins in a specific time period).

2 Coinjoin background

Bitcoin's UTXO-based transaction model [15] offers limited privacy, exposing recipients' addresses and transferred values. As outputs become inputs in subsequent transactions, heuristic techniques (e.g., common input ownership and script-type analysis [3]) can often reconstruct user activity. Attribution of a single address within a cluster reveals a user's financial history and approximate holdings. To counter this, users have adopted privacy-enhancing mixing protocols since 2011 [4, 9] to obscure input-output linkages.

Early mixing services such as Bitcoin Fog [9] and BitLaundry [4] required sending funds to a trusted coordinator, who batched and redistributed them with obfuscated outputs. These designs relied on full custody and mapping knowledge held by the coordinator, making them single points of trust. Collaborative transactions, coinjoins [12], were introduced to preserve user control without introducing custodial risk. Coinjoins contain inputs and outputs of multiple users, typically with repeated output values, so that, from on-chain observation, it is impossible to tell which combination of inputs and outputs is owned by a single user. Modern coinjoin implementations rely on trustless coordination that avoids custody while enabling the creation of such transactions. The coinjoin-specific terms are summarized by Table 1. This study focuses on the most widely deployed coinjoin variants: Wasabi 1.x (WW1) [27], Wasabi 2.x (WW2) [25], and Whirlpool (SW,AW) [19].

All these coinjoin designs execute the following steps:

A) Active wallets (clients) periodically connect to a coordinator API to determine the current coinjoin phase.
B) Wallets select coins and register them as coinjoin inputs at the coordinator and prove their ownership.
C) After the input registration ends, wallets confirm that they are still online and willing to participate.
D) Wallets register outputs, returning the value from own inputs to newly generated own addresses.
E) For SW, the coordinator prepares the final transaction and broadcasts it to the wallets for signing. For WW1 and WW2, transaction is deterministically created by all wallets locally from registered inputs and outputs supplied by coordinator.
F) Wallets verify the transaction and, if their included amounts are correct, sign (unlock) their inputs.
G) The coordinator collects all signatures and broadcasts the signed transaction to the Bitcoin network.
H) Wallets continue with step A). Once the transaction from step G) has been mined, coins and their anonymity metrics are computed.

If some of the wallets fail to confirm connection (step C), register outputs (step D), or unlock their inputs (step F), typically due to network connectivity issues, the coinjoin round is aborted (SW, WW1), or a smaller subset of active wallets is opportunistically formed (i.e., "blame round" for WW2). The offending inputs are removed and temporarily banned from subsequent participation

[1] https://github.com/crocs-muni/coinjoin-analysis
[2] https://crocs-muni.github.io/coinjoin/

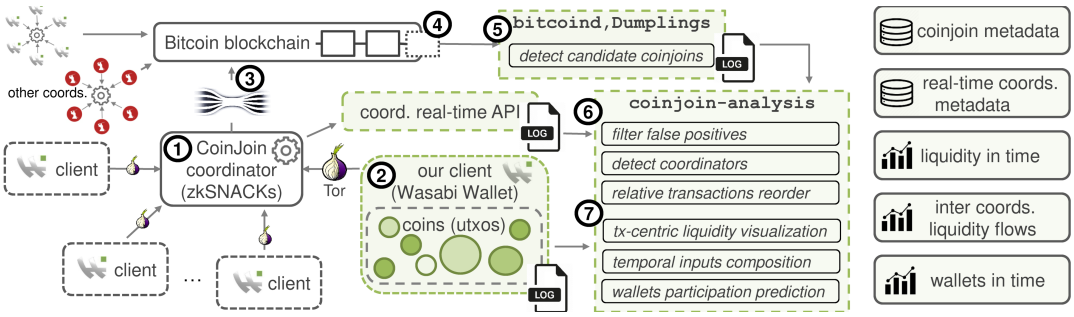

Figure 1: Overview of coinjoin creation ①–④ and coinjoin analysis ⑤–⑦ steps. The steps with a green background denote active data gathering and analysis performed in this paper.

for a defined time interval (i.e., "prison") as a DoS prevention mechanism. The mixing process adjusted to implementation specifics of different coinjoin designs is described in detail in Section C, with a step-by-step example for mixing 1 ₿ input. At a given time, only currently online and participating wallets are active participants contributing to the anonymity set of a given coinjoin transaction.

The coinjoin protocol steps A) to H) correspond to operations ①–④ on Figure 1, with the remaining operations ⑤–⑦ corresponding to the data analysis steps described in Section 3 and 4. The observed types of inputs and outputs are depicted on Figure 2.

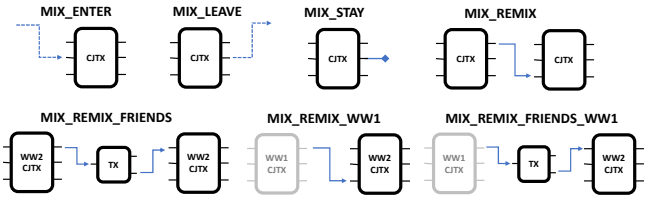

Figure 2: Input/output types observed in coinjoins. The upper row depicts three generic types of transactions (cjtx) valid for all three considered coinjoin designs, showing fresh liquidity (ENTER input), outgoing liquidity (LEAVE output), unmoved liquidity (STAY unspent output) and liquidity remixed in later transactions (REMIX). The bottom row presents special transaction types that avoided coordination fees ("friends-do-not-pay") in the WW2 zkSNACKs coordinator, where the input originates from an existing WW2 or WW1 mix user.

2.1 Trustless central coordinator designs

WW1 was the first operational implementation of the ZeroLink protocol [2]. After its release in 2018, WW1 supported only single-denomination coinjoins with a fixed output size of approximately 0.1 ₿. In 2019, version 1.1.0 introduced support for multiple output denominations (e.g., 0.2, 0.4, 0.8 ₿).

WW2 is an implementation of an improved protocol called WabiSabi [8], which allows for arbitrary output denominations, independently selected by each client based on all registered inputs. Both WW1 and WW2 initially operated under a single major coordinator, the zkSNACKs company [29], which developed Wasabi Wallet and the associated coordinator as open-source software. WW1 zkSNACKs coordinator was shut down on 19th May 2023 in favor of superseding the WW2 design. Following the voluntary shutdown of zkSNACKs WW2 coordinator on June 1, 2024 (citing legal uncertainty), multiple competing smaller WW2 coordinators were established, with at least four still operational.

SW is another implementation of the ZeroLink protocol. The coordinator, operated by the Samourai group, managed four distinct pools, with a specific output denomination: 0.001, 0.01, 0.05, 0.5 ₿ (denoted as 100k, 1M, 5M, 50M) and was shut down by FBI actions on 24th April, 2024. Initially, each pool conducted small coinjoins with a 5x5 input/output structure, later expanding to allow 8x8 transactions, all with the same pool-specific output denomination. Any fresh input is first split into outputs of the pool's standard denomination using a so-called *TX0* transaction. A new Whirlpool coordinator by the Ashigaru group was started on 31st May 2025 with two pool sizes of 0.025 and 0.25 ₿ (denoted as 2.5M and 25M).

Each design has a different fee structure, primarily consisting of a coordination fee (paid to the coordinator, might be 0%) and a mining fee (always paid to Bitcoin miners). Additionally, a client wallet may leave a (typically small) residual value that cannot be allocated to standard output denominations, which is then collected by the coordinator as a hidden "tip" (see Section G for more details).

2.2 Related Work

Stütz et al. [18] introduced methods for detecting WW1 and SW coinjoins and analyzed the Bitcoin blockchain up to February 2022, reporting counts of coinjoins, mixed volume, and an upper bound on mixing entities based on on-chain heuristics. Our work refines these estimates and extends them to WW1, WW2, and SW coinjoins, providing higher-resolution analysis that reveal the activity of larger wallets. We utilize existing coinjoin detection techniques [7, 17, 18, 20, 23], already included in Dumplings project [7], but apply additional post-processing: false positives detection based on expected connectivity between subsequent coinjoin transactions together with expected transaction flags and transaction reordering with respect to its default order given by mining time for less noisy analysis.

Wu et al. [26] proposed a generic model of mixing services and compared the mechanisms used by trusted-coordinator services with those used by WW1. They collected WW1 transactions via the public API and estimated the profits generated by coordination

fees through February 2020. Similarly, we collect public API data for post-zkSNACKs WW2 coordinators and utilize it for unknown coordinators detection.

Möser and Böhme [14] collected data on JoinMarket's offers by monitoring its IRC channel. They reported statistics on the amount of Bitcoins offered for mixing as well as the fees demanded for the use of the funds. Further, they used a heuristic for on-chain detection of JoinMarket transactions and compared the volume of the detected transactions with the volume assumed based on changes in the observed offers. Our work does not focus on JoinMarket-type of coinjoins without a centralized coordinator, but we utilize the basic idea of client-side participation to obtain better insight for on-chain data analysis.

Gavenda et al. [10] presented a generic approach for analyzing coinjoin transactions based on possible input-output mappings, demonstrating the infeasibility of this approach for large-enough WW2 transactions. Further, they showed that a large proportion of coinjoin outputs is spent in a privacy-harming manner.

The Dumplings project [7] provides baseline coinjoin transaction detection from on-chain data along with aggregate liquidity analysis up to November 2021. In our work, we incorporate Dumplings as a component of our pipeline, applying additional post-processing to reduce both false positives and false negatives in its outputs.

As information about the activity of different WW2 coordinators is crucial to users, several services providing basic data on current liquidity and completed coinjoin transactions emerged after the zkSNACKs coordinator shutdown. We are aware of two such currently running services, LiquiSabi [21] and Wabisator [1]. While these liquidity monitoring services focus on statistics on the current activity of the coordinators to facilitate informed selection of a current coordinator, we provide more detailed insights covering the whole eight-year history.

3 Client-side experiments

On-chain data provides only partial information about finalized coinjoins, reducing elaborate participant interactions only to the published transaction. In particular, all intermediate details known to a coordinator, such as the duration of the coinjoin orchestration, timing of input registrations, inputs that failed to sign the transaction, or precise time of transaction broadcast, are *not* preserved in on-chain data. Additionally, data visible only to the client, such as coin distribution, selected inputs and outputs, and changes in coin anonymity scores are also absent from on-chain records.

To obtain client-side insight missing from on-chain data, we conducted experiments using client wallets that actively participated in coinjoins across all three designs, covering different time periods and varying client configurations. By controlling the client wallet, we were able to capture temporal data from the client's perspective, offering a (partial) ground truth for the coinjoins in which we participated. This data is crucial for the estimation of the number of actively participating wallets and the related anonymity set.

We used *unmodified* client software wallets to participate in WW1, WW2 and SW coinjoins to prevent any unintentional modifications to the client's behavior. All implementations produce some level of logging into log files, in addition to the information displayed by the wallet graphical interface. Additionally, WW2 provides detailed information upon `listcoins` and `gethistory` RPC requests. Section J lists collected information in details. To protect the privacy of other users involved in the same coinjoins, we do not release the full dataset from these specific client-side experiments. Instead, we provide only aggregated results along with interpretive analysis.

3.1 Wasabi 2.x client experiments

The experiment was designed to capture the ground truth mapping of inputs to outputs, observe wallet behavior when targeting an anonymity score (*anonscore*) higher than the default and provide client-side characteristics which are not observable in the resulting on-chain data. As a result, we can later estimate the number and duration of participating wallets.

We conducted two experiments with the latest wallet versions available at the time: the first in May 2024 using the *zkSNACKs* coordinator with Wasabi Wallet 2.0.7.1 and target anonscore 25 (called *as25* experiment), and the second in March 2025 using the *kruw.io* coordinator with Wasabi Wallet 2.5.1 and target anonscore 38 (*as38*). The first experiment provided client-side insight while *zkSNACKs* was still operational, and the second confirmed these characteristics under the new dominant coordinator, *kruw.io*. The total cost spent on mining and coordination fees was roughly $600 for each experiment at a time: *as25* costing totally 1,003,653 sats (~4,047 sats/tx, 78.4% on mining fees) and *as38* costing 665,959 sats (~1,586 sats/tx, 62.7% on mining fees).

3.1.1 WW2 experiment design. The target anonscore of 25 was selected based on prior smaller experiments indicating that such an anonscore could be reached within one day. The target anonscore of 38 was set to be average between the lowest and highest value anonscore limits offered by the 2.5.1 version, yet also finishing within one day.

We manually set the target anonymity score (25 or 38), with all other wallet settings remaining at their defaults. Each iteration of the experiment (called *session*) began with the wallet loaded with a one coin (utxo) having an anonscore of 1. The wallet was allowed to automatically participate in coinjoins until all resulting output coins reached the target *anonscore*. If mixing did not complete within 24 hours, primarily due to coins being temporarily banned by the coordinator, the session was stopped. After each session, the wallet's coins (excluding temporarily banned ones) were consolidated into a single coin in an otherwise empty wallet before starting the next session. Throughout the process, all intermediate anonymity scores and client logs were recorded and analyzed.

For *as25* with *zkSNACKs* coordinator, we conducted 16 sessions with ~0.1 ₿ and 7 sessions with ~0.2 ₿ with identical wallet settings. In total, we participated in 206 coinjoins, resulting in 858 our coins with known input-output mappings. For *as38* with *kruw.io* coordinator, only sessions with ~0.1 ₿ initial liquidity were executed as no significant difference between 0.1 and 0.2 ₿ sessions from *as25* was observed. In total, we participated in 395 coinjoins, providing 1,955 coins with known input-output mappings for our wallet.

3.1.2 WW2 results. The results provide crucial insight into client-side behavior not otherwise retrievable from on-chain data. We use the insight to interpret on-chain data as described in Section 4.

The behavior observed in both *as25* and *as38* was remarkably similar with the following relevant differences: The average anonscore gain (increase of sum of anonscores over all wallet coins) per single coinjoin was 9.33 and 7.9 respectively. Achieving the target anonscore required on average 8.95 (as25) and 17.17 (as38) coinjoins[3] (see Figure 3). On average, our wallet participated only in half of all coinjoins created under *zkSNACKs* coordinator (*as25*) within the activity time period, while in almost all coinjoins for *kruw.io* (*as38*). The reason for the difference is that *zkSNACKs* started creation of a new coinjoin while the previous was finalizing (effectively two coinjoins in parallel), while *kruw.io* waited till finalization of the previous one. The stability of average anonscore gain (Figure 3) influences the expected number of coinjoins a wallet will actively participate in, before it reaches the desired anonscore target.

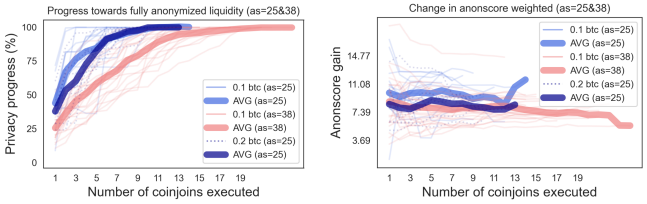

Figure 3: Privacy characteristics across consecutive coinjoin sessions targeting an anonscore of 25 for *zkSNACKs* (blue, 05/2024) and 38 for *kruw.io* (red, 03/2025). Thin lines correspond to individual sessions, while thick lines denote the average over sessions of the same type. The increase in *privacy progress* (left) reflects how long a wallet typically continues mixing before all coins reach the desired anonymity target. The anonscore gains per single coinjoin (right) show a comparable average anonymity gain across two tested coordinators, regardless of the initial wallet holdings and sequence of coinjoins in a given session.

The average number of inputs or outputs considered separately (Figure 4) is approximately the same for *as25* and *as38* experiments with 2.72 and 3.68 for inputs, and 4.15 and 4.95 for outputs. The average number of inputs is initially lower due to a small number of coins available in the wallet—starting each session with a single coin; but when enough output coins are available in the wallet (roughly 20 coins after around 10 coinjoins), it stabilizes as visible for *as38* in Figure 4. The average number of outputs has been consistent since the very first coinjoin in each session, independently of the number of coins in the wallet. However, the same does not hold for evaluating specific pairs of numbers for inputs and outputs. The *as38* experiment tends to participate with observably more inputs and outputs (Figure 5) due to a evolution of the wallet's coin selection algorithm[4]. We therefore use specific distribution (*as25*/*as38*) for prediction of the number of participating wallets under corresponding coordinator (Section 4.4).

[3]Influenced directly by the target itself, but also by the number of other participating wallets, which is estimated to be roughly the same for *as25* and *as38* (Section 4.4).
[4]CoinJoinCoinSelector.cs:SelectCoinsForRound()

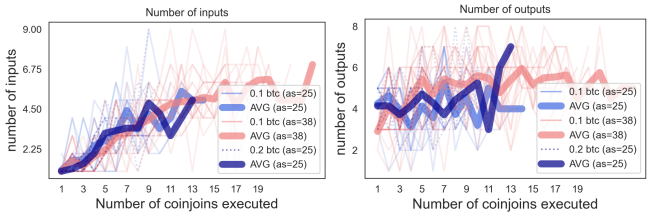

Figure 4: Number of registered inputs (left) and outputs (right) in progressing coinjoins, including average. The number of registered inputs initially grows with each coinjoin, as the wallet lacks sufficient coins available for registration, but eventually plateaus once enough partially mixed outputs have been generated through prior coinjoins. By contrast, the number of registered outputs remains relatively stable from the outset, being only mildly affected by the number of wallet coins, with only minor variations depending on the wallet's total holdings, used for the prediction of the number of participating wallets.

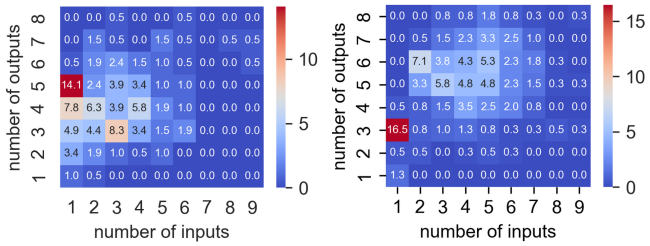

Figure 5: Frequency (%) of registered numbers of inputs and outputs for *zkSNACKs* (left, wallet version 2.0.7.1) and *kruw.io* (right, version 2.5.1). The latter typically involves more inputs and outputs, reflecting an evolution in the coin-selection algorithm and underscoring the importance of accounting for the expected client version when interpreting historical data for the estimated number of participating users.

The default anonscore at the time of *as25* experiment recommended by the Wasabi Wallet, Trezor Suite, and BTCPay Server in May 2024 was 5, and was increased to 10 at the time of *as38* experiment. Note that although our targets exceeded the wallet default, behavior at lower values remains analyzable.

For 21 out of 23 sessions of *as25* experiment, the anonscore of 5 was achieved for all outputs during the very first coinjoin of each session. This implies that if an anonscore of 5 was used instead of 25, only two sessions would continue mixing after their first coinjoin, producing observable remixed liquidity, while all others would stop and produce no remix liquidity at all[5]. On the contrary, no session out of 23 for *as38* reached anonscore 10 during its first coinjoin, always requiring at least two coinjoins and contributing with remix liquidity—making input to output linking more difficult for an attacker, as fully mixed output is at least two coinjoins away from its original fresh input.

[5]Wasabi Wallet v2.0.6 introduced a safety feature requiring at least two coinjoins for the first deposit into an empty wallet.

3.2 Wasabi 1.x client experiments

We used Wasabi Wallet 1.1.11.1 during June, September, and December 2020 to execute 7 coinjoin sessions (total 11 coinjoins) using default settings with anonymity set (*anonset*) target of 50. Note that WW1's and WW2's anonymity metrics are not directly comparable.

Input coins for coinjoin have to be selected manually and mixing does not start automatically. As any number of inputs can be registered into a single coinjoin, we registered 1-3 inputs to achieve the minimal value to form at least one output for the smallest (and most common) standard output denomination (around 0.1 ₿, slightly varying in time based on coordinator-side settings).

Mixed outputs tend to reach the default (=50) *anonset* threshold already after a single coinjoin. Out of seven separate mixing sessions, only two required a second, and only one a third coinjoin. Unlike WW2, only one standard-denomination output per wallet is allowed, with surplus value returned as change—yielding typically two outputs (std. denom. and change) per wallet.

As WW1 saw only limited activity, once WW2 became operational in mid-2022, more recent, extensive, and representative experiments were no longer possible.

3.3 Whirlpool client experiments

For SW, the number of participating client wallets in a single coinjoin is known and equal to the number of coinjoin's inputs; the SW wallet will register only a single own input per coinjoin. The subsequent remixes are free of charge (including the corresponding mining fee paid by fresh inputs coming from *TX*0). The active wallets are queuing a subset of their already mixed coins for these free remixes, occasionally being selected randomly by a coordinator for participation in coinjoins. To estimate the number of wallets active at a given time (approximating the potential anonymity set), we rely on three parameters: (1) the queue length of coins awaiting remix, (2) the fraction of our inputs selected by a wallet, and (3) the probability of being chosen for participation. Through prolonged participation with our wallet, we directly measure (2) and (3); combined with observed coinjoins and their remixed inputs, we can infer (1)—the remix queue length.

We used the SW client embedded in Sparrow Wallet [24] for 47 days from November to December 2021 (Sparrow version 1.5.2, SW version 0.23.27) and for 55 days from July to September 2023 (Sparrow version 1.7.7, SW version 0.23.36) with 100k (45 executed coinjoins), 1M (160 coinjoins) and 5M (44 coinjoins) pools at Samourai coordinator (Table 2 in Section H). The cost of all experiments combined was roughly ~$130 paid on mining (~20.9%) and coordination (~79.1%) fees.

In total, 249 coinjoin transactions were executed in seven non-overlapping time intervals, always starting with its own new funding transaction *TX*0 providing input coins for a given pool. The first mix for each coin happens quickly, as the coordinator waits only for one fresh coin from another wallet, the rest being randomly selected from coins in the free remix queue. After a wallet observes the announcement of a new potential coinjoin transaction by the coordinator, it registers one randomly selected coin with the same denomination into the free remix waiting queue. As a result, the probability of participation in a coinjoin as a free remix input shall correspond to the ratio of the number of free remixes our wallet participated in to the total number of free remix inputs in all coinjoins executed during the same period. We observed the following minimum/maximum queue lengths: 589/1180 (100k pool, 2023), 146/340 (1M pool, 2021), 355/355 (1M pool, 2023), and 172/282 (5M pool, 2023), corresponding to the estimated anonymity set of other active wallets, with more details in Section H.

3.4 Active querying of coordinators API

We collected a ground-truth dataset of coinjoin transactions mapped to their coordinator using two independent methods: 1. periodic queries to *wasabist.io* and *wabisator.com* aggregators, and 2. direct queries to the list of known coordinators via their coordination API (`human-monitor` and `status` commands). Due to temporal outages on the aggregators and our collection sides, none of the collected lists are fully complete. Figure 6 shows the daily number of transactions available in different datasets, together with the number of transactions initially *not* attributed by any ground-truth mapping (red area, 6.88%). The notable gap is from May to early July, before public monitors and our active API monitoring were fully operational, and then occasional collection outages.

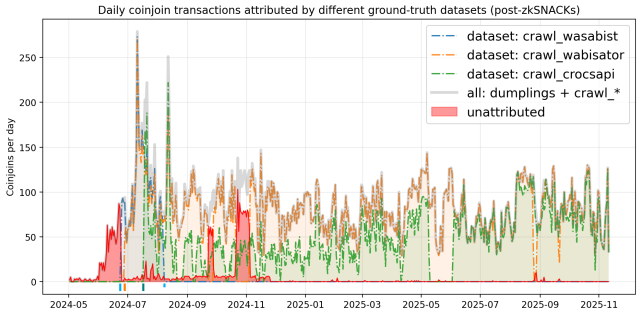

Figure 6: Coverage of the crawled dataset with ground-truth mappings of coinjoin transactions to their coordinators. The *crawl_wasabist* dataset (blue) represents the earliest coordinator monitoring, but ended in August 2024. The *crawl_wabisator* dataset (orange) began shortly thereafter and contributes the majority of mapped transactions, remaining active with occasional outages. The *crawl_crocsapi* dataset (green) offers detailed coordinator statistics collected at one-minute intervals, but until June 2025, only for selected coordinators. The red areas mark periods of coinjoin activity without corresponding ground-truth mappings—outages and activity of unknown coordinator(s).

The original zkSNACKs coordinator also provided an API to obtain TXIDs of finalized transactions currently waiting in the mempool. However, it is no longer provided by the new coordinators, but it can still be reliably inferred from the time and the composition of inputs. No public API is now running for WW1.

The aggregated ground-truth dataset with mappings used for evaluation is created using transactions, for which at least one partial dataset provides a mapping. When combined with all coinjoin-like transactions extracted from blockchain using Dumplings (only post-zkSNACKs transactions are considered), the difference (red)

highlights transactions with missing attribution to their coordinator—the target of our coordinator attribution algorithm (Section 4.2.1). The list of collected ground-truth mapping of transactions to coordinators is made public.

4 On-chain data analysis

We analyze *all* historical coinjoin transactions created using Wasabi 1.x (WW1), Wasabi 2.x (WW2), and Whirlpool (SW), extracted fromthe Bitcoin blockchain. We use additional insight obtained by active participation in coinjoin rounds to observe wallets' behaviors and gather empirically observed parameters described in Section 3.

The time analysis period starts with the first ZeroLink-based coinjoin 2018-07-19 (WW1) and ends with the recent cutoff time 10th November 2025 before this paper finalization.

Given the extensive dataset—encompassing over half a million coinjoins over an 8-year period—and the numerous visualizations generated from this analysis, we focus on presenting only the results relevant to our research questions (RQs) and illustrating the analysis methodology used. Additional results are presented in the Appendix and online[6].

4.1 Methodology

All finalized coinjoin transactions are permanently recorded on the Bitcoin blockchain, allowing for retrospective extraction and analysis of historical trends and events of privacy mixes. We use three base projects in the analysis pipeline:

- ***bitcoind*** [16] for retrieving Bitcoin blockchain;
- ***Dumplings*** [7] for detection of coinjoins;
- ***coinjoin-analysis*** project (newly developed) for all subsequent analysis of coinjoin data.

For all three coinjoin designs, the following processing steps are performed as shown in Figure 1 (steps ⑤-⑦):

(1) **Discovery of candidate coinjoin transactions ⑤:** usage of Dumplings tool [7] to heuristically detect and extract on-chain coinjoins.
(2) **Rectification of misclassifications and misplacement ⑥:** removal of false positives (non-coinjoin transactions), misclassifications (postmix spent instead of coinjoin transactions) and misplacement (relative disordering of transactions due to mining fee spikes).
(3) **Attribution to coordinator ⑥:** detection of the coordinator of a given coinjoin based on active monitoring and retrospective heuristic attribution (Section 4.2.1).
(4) **Extraction of metadata from coinjoin series ⑥:** detection of liquidity events (amounts and numbers of coins entering, staying in, or leaving given mix pool), fees paid, and retrieval of system parameters over time.
(5) **Analysis and visualization of trends ⑦:** Visualized global trends in time and detection of notable events.

Additionally, we independently implemented a subset of the functionality—coinjoin detection and basic liquidity statistics—into the BlockSci project[7] (C++) for basic validation of the results obtained. While BlockSci provides superior analysis performance (seconds to detect coinjoins after an initial blockchain synchronization when 256 CPU cores were used), it also requires a machine with at least 500GB RAM due to in-memory blockchain representation. The *coinjoin-analysis* (written in Python) performs the computation while requiring less than 100GB of RAM. The processing for both *coinjoin-analysis* and BlockSci highly benefits from additional CPU cores due to internal tasks parallelization. No GPU acceleration is used.

4.2 Coinjoin pre-processing algorithms

4.2.1 Coinjoin discovery algorithm. For WW1 and SW, we applied the default coinjoin discovery from the Dumplings project, based on Stütz et al. [18]. The algorithm leverages knowledge of output denominations, expected input–output counts, limited address reuse, and predefined initial pool transactions, followed by graph traversal (for SW).

For WW2, we decreased the required minimal number of inputs from 50 (Dumplings' default) to 15 to account for smaller transactions created by new coordinators after zkSNACKs was stopped (May 2024). The resulting increase in the number of false positives is mitigated by a subsequent false positives detection algorithm.

The Dumplings project also provides lists of pre-mix (transactions immediately preceding a coinjoin) and post-mix (immediately following) transactions. In practice, we observed that some genuine coinjoins—particularly in SW—were misclassified. To address this, we reprocess all pre- and post-mix transactions using coinjoin detection rules and reassign misplaced ones to the coinjoin set.

As the initial set of coinjoins is a crucial input for subsequent analyses, we also re-implemented the mentioned coinjoin detection into the BlockSci project for an independent verification of results. After comparison with a list of all finished coinjoins obtained by actively querying running coinjoin coordinators, we obtained a perfect match for SW transactions and a near-perfect match for WW1 and WW2 transactions. All differences were manually identified as misclassifications of the Dumplings project (all discovered by false positive detection) or (rare) cases of completed coinjoin transactions announced by a coordinator, but never recorded to the blockchain ledger. The latter case happens when one or more inputs are double-spent before a finalized coinjoin transaction was mined into a block—typically due to a transaction stuck in the mempool during a sudden mining fee increase and the original input owner(s) decided to move their funds earlier.

4.2.2 False positives removal. We filter transactions returned by the coinjoin discovery algorithm using two false positives detection rules based on additional context of coinjoin transactions not utilized by the discovery algorithm:

1) Missing links to other coinjoins: All three designs rely on remixing to expand the anonymity set, so links to adjacent coinjoins—either as inputs or outputs—are expected. The only legitimate exceptions are initial coinjoins (with no input linkage) and final ones (with no output linkage) within a given coordinator's active period.

[6] https://crocs-muni.github.io/coinjoin/

[7] https://github.com/crocs-muni/blocksci

Transactions lacking any such connections are very likely to be false positives.

2) Excessive number of reused addresses: Usage of the same locking script (address) directly reveals ownership by the same entity and is therefore intentionally avoided by all coinjoin wallets, except for occasional implementation bugs. No or almost no address reuse is therefore expected and transactions with more than 30% of address reuse are almost certainly false positives.

3) Unexpected transaction properties: if coinjoin-like transactions exhibit flags or formats not seen in transactions produced by known coinjoin software, we analyze them and mark them as false positives. In particular, we excluded all transactions with single non-standard denomination change output, RBF enabled and no Taproot outputs for WW2.

For WW1, WW2 and SW, we detected 402, 4026 and 0 (zero) false positives, respectively. The SW detection likely contains no false positives due to very specific structure of its initial splitting transaction *TX0* (outputs corresponding to pool size, OP_RETURN output), fixed structure of its coinjoins (same number of inputs and outputs, same value of all outputs equal to pool size) and remix connections to other existing coinjoins; all utilized by the default discovery algorithm.

The decreased threshold (down to 15) for a number of inputs for WW2 naturally resulted in an increased number of false positives, which we mitigated by detecting candidates with a missing connection to any other coinjoin or exhibiting a high level of address reuse. We manually verified all the candidate false positives and continue to add them daily to an exclusion list.

Note that in all cases, an active attacker might intentionally craft a transaction looking exactly the same as a genuine coinjoin, potentially even using outputs from real coinjoins to confuse detection algorithms used. One possible motivation to do so might be to anticipate a future exclusion of any coinjoin transaction from address clustering performed by chain analysis companies due to broken assumptions behind *common input ownership* heuristic. However, creating a coinjoin-looking transaction outside a set of transactions finalized under any known coordinator (obtainable by active monitoring of its API) will make such a transaction distinctive. We are not aware of any such suspicious transaction, but the required API data is missing retrospectively for certain time periods.

4.2.3 Retrospective relative transaction reordering. Most coinjoin designs require that an output from a prior coinjoin receive at least one block confirmation before it can be used. Although querying the coordinator's API at broadcast time would allow precise ordering of coinjoins, such data is unavailable to us retrospectively. While each block includes a timestamp, relying on it for transaction ordering is problematic: (1) multiple coinjoins may be included in the same block despite being broadcast at different times, and (2) a coinjoin broadcast earlier may appear in a later block due to underestimated mining fees. These inconsistencies introduce noise into analyses, especially when fees spike abruptly, delaying transaction confirmation and dispersing related coinjoins across multiple blocks. This degrades metrics such as "burn time" (time till a coin is remixed) and mix value flows tracking.

For WW2, which features a high remix ratio (typically exceeding 90%), we developed a relative reordering algorithm to compensate for the absence of reliable broadcast timestamps. The algorithm is not applicable to SW and only marginally to WW1, due to their low remix rates and sparser transaction linkages.

To obtain a baseline for evaluation, we performed active monitoring of the WW2 zkSNACKs coordinator (as described in Section 3.4). We collected TXIDs and times of appearance in mempool for coinjoins created from May 2024 until the zkSNACKs' shutdown on June 1st. During this period with stable mining fee, 94.95% of 1129 coinjoins were confirmed within 12 blocks (roughly 2 hours) of broadcast—which we use as a threshold below which we do not reorder transactions. We also do not reorder transactions with a difference bigger than 2 weeks, which is unlikely to be caused by a delay in mining, but is typically caused by the start of a new coordinator instead.

The reordering process then involves the following steps:

(1) For each coinjoin cj, estimate its timestamp $t(cj)$ as the mining time of its block.
(2) Construct a partial order $\prec$ on coinjoins based on input-output dependencies: if cj_m consumes an output from cj_n, then $cj_n \prec cj_m$. The relation also includes all pairs implied by transitivity.
(3) For each $cj_n \prec cj_m$, such that $t(cj_n) > t(cj_m)$ and 2-hours $< t(cj_n) - t(cj_m) <$ 2-weeks, set $t(cj_n) = t(cj_m)$.
(4) Estimate burn time of output of cj_m spent in cj_n as the size of the longest chain from cj_m to cj_n, rather than absolute block height differences.

This method significantly improves temporal analysis, as illustrated in Figure 7, by the removal of periods with artificially high burn times, and by a tighter collocation of fresh liquidity inputs originating from the same wallet.

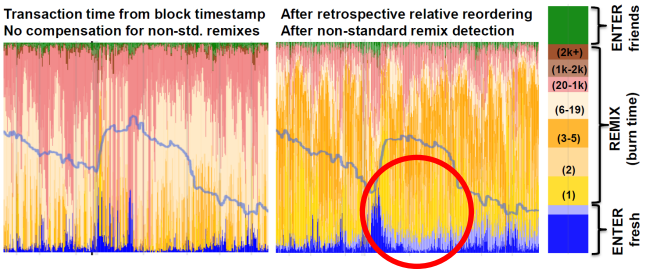

Figure 7: Impact of post-processing and temporal inputs composition of the same time period. Each vertical bar corresponds to one single coinjoin transaction with sub-bars color-coded according to Section 4.3.1. The fresh input patterns (blue, green) and burn time patterns (gold to brown) are significantly improved, with more accurate resulting Interim pool's liquidity increase (blue line, Sect.4.3). The red circle marks the period when a large wallet introduced fresh liquidity over several consecutive coinjoin transactions, producing a temporary rise in *Interim liquidity* followed by a decline once the coins were mixed and spent. Without post-processing (left), the visualization fails to correctly capture both the liquidity inflow time and its remix duration.

4.2.4 Attribution of coinjoins to their coordinators. The core challenge lies in accurately mapping coinjoin transactions to their coordinators if multiple were running in parallel for the same coinjoin design. While active querying of a coordinator's API can definitively attribute transactions, such querying is only possible in real-time and cannot be applied retrospectively. Furthermore, some coordinators are not publicly known, precluding any direct queries altogether. This task is now particularly relevant for WW2, as its client wallet allows users to easily switch between coordinators.

We employ a heuristic based on remix behavior. Using analysis of inter-coordinator remix rate for transactions with known ground-truth mapping to their coordinators (see Section D.1), we established that a substantial proportion—typically at least 40%—of remixed outputs are quickly reentering into coinjoins coordinated by the same operator as shown on Figure 13. Thus, if a coinjoin includes a dominant fraction of remixed inputs previously associated with a known coordinator, it can be attributed to that same coordinator. In scenarios where a coordinator ceases operation, previously linked wallets stop mixing until users eventually (not immediately) transition to a different coordinator by manually reconfiguring their wallets.

Finally, coinjoin sequences attributed to the same coordinator are merged, accounting for possible inactivity before or after each sequence. This temporal context refines attribution by distinguishing separate coordinators from temporary disruptions. The detailed rationale for parameters selection and analysis of algorithm's sensitivity to missing attributions is provided in Section D.

Using the described methodology, we attributed early coinjoin transactions for a range of known smaller coordinators before active API monitoring (see Section 3.4) was started. We also found additional, previously unknown coordinators, which operated only in 2024 and produced the earliest transaction from all post-zkSNACKs coordinators (denoted as *unknown_2024_e85631 / 28ce7b* in Table 3). See Section E for more details on their properties.

RQ1

RQ1: *Which unknown coordination services can be retrospectively identified using observable transaction behavior?*

The difference between transactions gathered by API monitoring from all known coordinators and all coinjoin transactions recorded on blockchain provides candidates for ones from unknown coordinators; a feature suitable both for retrospective as well as, almost real-time, future detection. When combined with the attribution algorithm (Section 4.2.4), we found two distinct clusters of transactions by past unknown coordinators operational between May and November 2024 (see Section E), mixing 46.89 ₿ and exhibiting practically no remix with any other coordinator; behavior which would be consistent with a non-public coordinator.

4.3 Coinjoin visualization algorithms

Preprocessed data, using the techniques described in Section 4.2, enable analysis of coinjoin-wide ecosystem dynamics, coordinator-specific behaviors, and notable observable events, with the goal of visualizing coinjoins over time to facilitate inspection and identification of temporal trends.

All Bitcoin transactions, including coinjoins under the same coordinator (mix, pool), can in principle be represented as a directed acyclic graph (DAG) with transactions being nodes and outputs used as inputs being edges. However, due to the significant number of coinjoin transactions executed, each potentially involving hundreds of inputs and outputs, naive DAG visualization is not beneficial for the analysis of temporal events.

As the main goal of coinjoin is to obfuscate the link between users' fresh inputs entering and outputs leaving it, we primarily focus on visualizations capable of capturing users' funds entering and leaving the pool.

4.3.1 Temporal inputs composition. To visualize coinjoins sequentially in time, we utilize a stacked bar graph where each coinjoin corresponds to one bar stacked from sub-bars capturing inputs of the same type (Figure 2) and visualized with a specific color. We use the following coloring scheme for inputs as shown on Figure 7: fresh MIX_ENTER (blue), postponed[8] fresh liquidity (light blue), remixed (colors from gold to sienna based on a remix burn time with darker hue for higher burn times) and remix from "friends"[9] (green). The corresponding sub-bars are stacked in the same order from the bottom of a graph with fresh inputs at the bottom, remixes in the middle and "friends" inputs at the top.

The overall height of the stacked bar corresponds to a combination of all inputs of the same type—either the combined value in ₿ or a number of such inputs. Such a visualization captures accurately temporal changes in the absolute value (*notnorm*) or number of inputs for each separate coinjoin, but does not allow easy comparison of patterns formed by intermediate sub-bars due to uneven height of bars. We therefore create also normalized (norm) versions of these two graphs, resulting in four primary versions: *value&notnorm* (suitable for spotting patterns related to absolute input value), *value&norm* (remix patterns related to value), *numbers&notnorm* (patterns of absolute input numbers) and *numbers&norm* (remix patterns related to number of inputs). The Figure 7 shows a typical case for the normalized number of inputs for WW2. Figure 8 captures a wallet's WW1 mixing session duration and the correlation between its fresh and mixed coins.

4.3.2 Transaction-centric liquidity visualization. Past studies [7, 14, 18, 23] have estimated the amount of liquidity entering coinjoin pools by measuring the value of fresh inputs in detected coinjoin transactions. While this method captures the initial inflow of funds, it offers limited insight into the subsequent behavior of those inputs and tends to overestimate the actual value that was effectively mixed. For instance, if a coinjoin includes a very large input that exits immediately as a similarly large, easily distinguishable output, this non-standard output likely retains little to no anonymity and can be excluded from further mixing analysis due to its minimal anonymity set. To address this limitation, we propose a transaction-centric liquidity visualization approach that tracks not only the initial entry of funds into the pool but also the ongoing behavior of

[8] A fresh input may not be completely mixed into standard denomination outputs in single coinjoin due to its (larger) size, becoming a non-standard output and entering mix again in subsequent coinjoin.

[9] Inputs to WW2 not paying zkSNACKs coordination fee as they are coming directly from WW1 or other WW2 user.

wallets managing that liquidity. This enables a more accurate and nuanced understanding of real mixing activity over time.

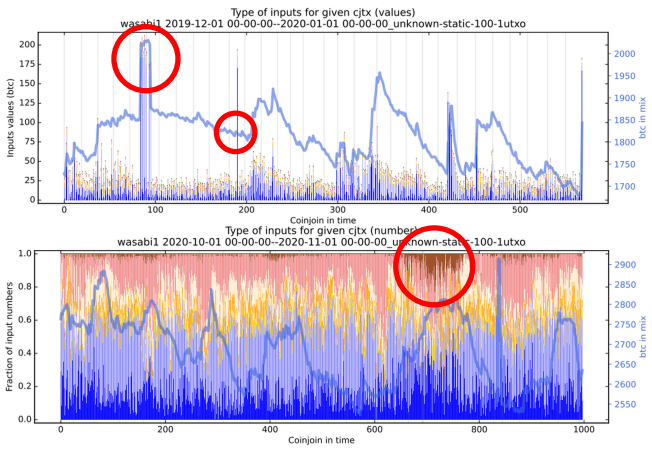

Figure 8: Visual detection of large wallets behavior in WW1. The upper sub-graph (*values&notnorm*) detects the length of mixing and subsequent spending of mixed outputs where interim liquidity decreases down to the initial level (coinjoins around 90, highlighted by a red circle) or input without further mixing with no impact on pools liquidity (coinjoin 189). The lower one (*numbers&norm*) captures correlation of fresh inputs (blue) with a significant influx of older ones (brown) possibly coming from the same wallet (coinjoins around 650–750). The remaining two variants (*numbers&notnorm* and *values&norm*, not shown) are typically suitable for inspection when an excess number of small inputs are present.

Inputs and outputs of each coinjoin transaction can be interpreted as updates to the overall liquidity state of the shared mixing pool under a given coordinator. Specifically, a previously unmixed input (MIX_ENTER) represents an inflow of fresh liquidity, increasing the pool's total value. Conversely, any output subsequently spent outside of a coinjoin (MIX_LEAVE) reduces the pool's liquidity. Inputs originating from prior coinjoins, or outputs consumed directly by later ones (MIX_REMIX) are considered internal recycling of liquidity and therefore do not alter the pool's net value.

For each coinjoin transaction cj_i in the chronologically ordered dataset, we compute the following metrics:

- **Interim pool's liquidity**: The difference between the sum of all MIX_ENTER inputs and the sum of MIX_LEAVE outputs of all transactions up to cj_i. The included liquidity is computed based on outputs with standard denominations (value appearing at least two times for a given coinjoin) to account for distinctive outputs easily separable from other mixed outputs.
- **Unmoved outputs ($MIX\_STAY(today)$):** The cumulative sum of outputs of coinjoins that have not been spent up to cj_i by any transaction till the current day. Note that this value is not stable for future re-computations and will naturally decline over time as users eventually spend their coins. Therefore, we also introduce a temporarily unmoved outputs metric. The variation $MIX\_STAY(1m)$ calculates the total value of outputs that have remained unspent for at least one month after their creation, but may have been spent later.
- **Actively remixed liquidity:** Calculated as the difference between the interim pool liquidity and the temporarily unmoved outputs (MIX_STAY(1m)). This reflects the volume of liquidity actively participating in the mixing process in the specific time-frame.
- **Remix rate:** The proportion of remixed inputs (numbers or values) with standard denominations to all other types of inputs.

These metrics allow us to visualize the temporal impact of individual coinjoin transactions, capturing meaningful shifts in the pool's state. Such visualizations can reveal the distinct behavioral signatures of specific wallets, as demonstrated for all three designs in Figure 9 and in more detail in the Appendix. The pseudocode for their computation is provided in Section K.

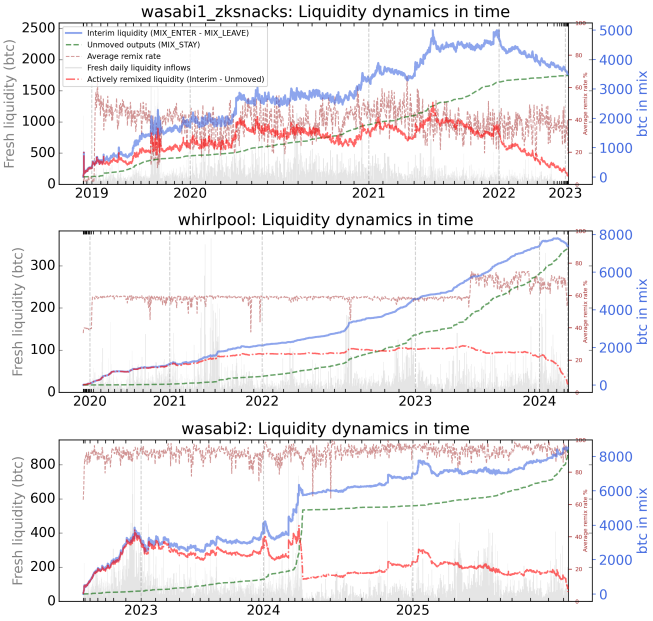

Figure 9: Cumulative liquidity dynamics for studied coinjoin designs. All WW2 and SW pools are merged together, respectively. The x-axis is proportional to the number of coinjoins executed in a given period, with dashed vertical lines separating years and small ticks months. Aggregated fresh daily liquidity entering the mix is plotted in gray vertical bars.

Note that when analysis of post-zkSNACKs coordinators is separately performed, the Interim pool's liquidity can become temporarily negative, as inputs previously mixed by other coordinators (MIX_REMIX_FRIENDS) are not counted as fresh inflows, but outflows are (as we cannot pair mixed outputs to their inputs). Depending on an analyst's needs, the computation can be kept unchanged (with an interpretation of the coordinator temporarily losing more liquidity than its completely fresh inputs), or MIX_REMIX_FRIENDS inputs can be treated as fresh liquidity (MIX_ENTER) with the results recomputed. Both results are relevant depending on the circumstances.

4.4 Number of participating wallets

Estimating the actual number of distinct wallets participating in a given coinjoin transaction is inherently difficult due to design choices that intentionally obfuscate participant identities. For example, the wallets typically register each input and output under a separate Tor identity to prevent linkage based on IP address.

However, the total number of inputs and outputs for each coinjoin is publicly observable from on-chain data, creating an upper limit on the number of wallets involved being equal to the minimum from a number of inputs and outputs (typically inputs being smaller). But as documented by client-side experiments (Section 3), such an upper limit is certainly too high, as a single wallet typically registers more than one input and output into a given coinjoin. For example, 4.95 outputs were registered with *kruw.io* on average.

Although the specific contribution (in terms of the number of inputs or outputs) from each wallet is inherently unknown, the total input count in a coinjoin itself already represents an aggregate of all participating wallets' contributions. The wallet's coin selection algorithm, influenced by factors such as the number of coins held and their anonymity scores, generates a particular distribution of number of inputs and outputs. The overall distribution of a coinjoin can be understood as the aggregation of the input distributions of participating wallets. If all wallets were in the same internal state, this distribution would approximate a simple average; in practice, differing wallet states introduce heterogeneity into the aggregate.

We used the real distributions of *number of outputs* obtained during client-side experiments (Figure 4) to predict the number of wallets and estimate the prediction's error bounds. The resulting bounds for both client-side experiments provide reasonably tight and usable estimates, tighter for *as38*. For typical coinjoin with 300 outputs, the point estimate for the number of wallets is N=60.6 wallets, 95% CI [55.4, 65.8] based on Wald CI via the delta method [11] using the *as38* distribution. See Section F for detailed evaluation.

The model predicts roughly 60 active participants (after smoothing variation across subsequent coinjoins) for most of the *zkSNACKs* period, with *kruw.io* initially starting below 20, gradually rising above the *zkSNACKs*'s level to 70+ as seen on Figure 10.

4.4.1 Local mismatch between inputs and outputs. One can also use a prediction based on the number of coinjoin inputs rather than outputs; the predicted number shall equal the prediction based on outputs. We propose an approach to estimate the average number of wallets per coinjoin using a sliding window method. First, we take the observed average number of *outputs* (e.g., 4.95 for WW2 kruw.io) per wallet as a stable reference. Then, for each interval (e.g., monthly), we compute a synthetic average number of *inputs* per wallet such that the number of estimated wallets derived from inputs equals the number derived from outputs[10]. Specifically, we calculate this synthetic input/output-to-wallet ratio by minimizing the L1 (Manhattan) distance between the two resulting wallet number estimates over the interval. While the method could be anchored in either inputs or outputs, we select outputs due to their lower variance, as observed by our client-side experiments (Section 3).

[10]The standard coinjoin only resends a wallet's input value back to itself. The exceptions are one additional wallet for outputs belonging to a coordinator for collection of fees and a recently introduced feature pay-in-coinjoin of WW2, which is still experimental and not directly available in the WW2 GUI.

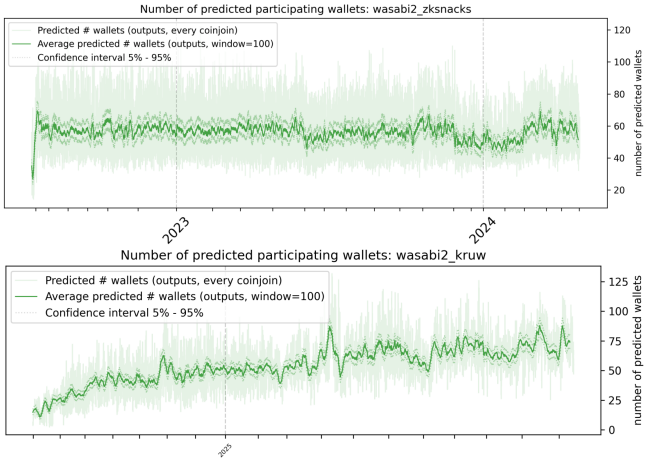

Figure 10: The output-based predictions of the number of participating wallets for WW2 *zkSNACKs* (top) and *kruw.io* (bottom) with a smoothed average and error bounds (dashed, CI 95%). The *zkSNACKs* coordinator consistently involved roughly 60 participating wallets during its operational period, while *kruw.io* began with fewer during the post-zkSNACKs competition, then gradually increased and even surpassed this number as it gained dominance.

Although we can use the real distribution of *number of inputs* from *as25* and *as38* experiments, the design of the experiment, starting with a single coin in a wallet, artificially limits the number of registered coins until a wallet is populated with enough coins available for selection. Experimentally, we verified that dropping the first three (for *as38*) or first five (for *as25*) coinjoins from each session is enough to obtain predictions matching the one based on the number of outputs (Figure 21).

A significant divergence between the wallet count estimates derived independently from inputs and outputs—despite the (inherently unknown) ground truth for these values being equal—indicates the presence of wallets with coin distributions that temporarily deviate markedly from the average. To account for possible changes in wallet coin selection algorithms over time, the synthetic ratio is computed separately for a selected interval (e.g., 10 neighboring coinjoins or a month) rather than as a single static value.

RQ2

RQ2: *How can temporal characteristics of coinjoin usage be estimated from on-chain data?*

The combination of ground-truth from our client-side experiments with retrospective data recorded on-chain (Section 4.4) allowed us to predict the total number of wallets participating (effective anonymity set) in time for WW1 and WW2 zkSNACKs coordinators, being long-term roughly around 60 wallets on average for both. The post-zkSNACKs coordinator *kruw.io* gradually raised from below 20 to between 60-80 concurrent wallets as of November 2025. The error bounds are usably tight, e.g., for the typical *kruw.io* coinjoin with 300 outputs, the estimate is 60.6 ± 5.2 wallets for 95% Wald CI.
For WW1 and WW2, transaction reordering and transaction-centric liquidity visualization (Section 4.3.2) clearly reveals inflows and outflows when a wallet's inputs exceed those of other participants by roughly an order of magnitude. In such cases, the mix fails to conceal how long the inputs remained in mixing or whether they were spent or left unspent. The privacy impact is particularly pronounced for smaller coordinators, such as *opencoordinator*, where mixing durations for wallets with inputs as small as 1–2 ₿ are readily observable.

5 Ecosystem insights

Using the toolchain described in Section 4, we extract available on-chain information from coinjoins of WW1, WW2 and SW from 19th July 2018 (start of WW1) until the 10th November 2025 (cutoff date for this paper), with a summary presented in Table 3. The time period can be divided into three primary periods—before the shutdown of zkSNACKs's WW1 coordinator, before the shutdown of Samourai's Whirlpool and zkSNACKs's WW2 coordinators (April/May 2024) and the period after, with multiple new post-zkSNACKs WW2 coordinators operating. These periods slightly overlap as zkSNACKs WW1 and WW2 coordinators were operated roughly a year in parallel and post-zkSNACKs coordinators were partially operational already in May 2024. Finally, post-Samourai Whirlpool coordinator Ashigaru is operating since 31st May, 2025 with very little activity so far. The complete results for monthly intervals are provided in the Section M, N and O.

Collectively, all pools mixed over 391,564 ₿ of fresh input liquidity[11], distributed across 1.93 million inputs (both fresh and remixed). Both WW1 and WW2 (under zkSNACKs coordinators) generated approximately 35,000 transactions each, typically involving low hundreds of inputs and outputs. All post-zkSNACKs coordinators for WW2 collectively added another 42,717 transactions. In contrast, SW produced an order of magnitude more transactions (541,131), but primarily consisting of small transactions with exactly five inputs and five outputs or slightly larger with eight inputs and eight outputs. Recently started Ashigaru coordinator for Whirlpool coordinated only 287 coinjoins in its 2.5M and 25M pools so far.

5.1 Aftermath of 04-05/2024 shutdown

On 24th April 2024, all four pools under the Samourai SW coordinator, as well as related infrastructure, were forcefully shut down by an FBI operation [22]. One month later, zkSNACKs, the company behind the WW1 and WW2 coordinators, decided to shut down operations as well, citing legal uncertainties [28].

If a user of the WW1 wallet upgraded to the WW2 wallet with the zkSNACKs coordinator (before its shutdown on 1st June, 2024), the wallet's coins formerly mixed with the WW1 coordinator were fee-discounted also for the WW2 coordinator. Simple selection of an alternative WW2 coordinator (post-zkSNACKs) became available from version 2.0.8 (Hydra) released on 1st June, 2024 (same day when zkSNACKs coordinator was stopped), allowing for competition[12] of alternative coordinators for users and liquidity.

The WW2 zkSNACKs coordinator enforced a minimum and maximum of inputs and outputs at 150 and 400, respectively (except for the first month of its operation). Except *kruw.io*, all other post-zkSNACKs coordinators currently operate with significantly lower minimums (e.g., 10-50).

On average, anonymity increases with more participants. The coordinators with a higher rate of coinjoins and larger liquidity, taken as a heuristic proxy for the (unknown) number of real participants, attract more new users, naturally leading to liquidity centralization mainly with one coordinator. Indeed, until the shutdowns of all three main coordinators in 2023 (WW1) and 2024 (SW, WW2), no other relevant coordinators emerged for a given design despite open-source availability and clear economic incentives in collected coordination fees. Around three months after shutdown (August 2024), *kruw.io* clearly emerged as a dominant coordinator for WW2 with two orders of magnitude more liquidity than other remaining coordinators and kept its dominance till the cutoff date (10th November 2025). The number of its coinjoins is gradually increasing, now typically over 800 per month, though still lower than zkSNACKs' maximum of almost 2,500. Surprisingly, both *gingerwallet* and *opencoordinator* kept a comparable number of overall coinjoins as *kruw.io*, but with an order of magnitude lower number of inputs and two orders of magnitude lower fresh input values.

The Figure 12 shows direct liquidity flows between WW2 coordinators. Such a flow is a result of a wallet owner manually switching to a new WW2 coordinator (e.g., from *zkSNACKs* to *kruw.io*), with outputs already mixed under the previous coordinator being remixed again as inputs under the new one. The majority of outflows from zkSNACKs coordinator entered *kruw.io* (85.1%, 635.9 ₿). Notably, a significant liquidity initially mixed by other coordinators than *kruw.io* was later remixed again with *kruw.io*—mainly from *wasabicoordinator* (121.4 ₿) and *opencoordinator* (119.5 ₿)—as users converge to the dominant coordinator or use more than one.

[11] We exclude non-mixed outflows (for WW1) and inflows entering WW2 from WW1 or sent between WW2 users. Repeated remixes are also not counted.

[12] See https://liquisabi.com/ for statistics on current coordinators.

RQ3

RQ3: *How can the popularity, liquidity, and transaction frequency be retrospectively measured?*

Retrospective activity measurement combines coinjoin detection, false positives filtering, attribution to coordinators, transaction reordering, and transaction-centric liquidity analysis to provide new insights and more precise liquidity measurements. The WW1, WW2, and SW all exhibit markedly different aggregated behavior of their users. The coinjoin pools have collectively mixed more than 391,564 ₿ of fresh liquidity, distributed across 1.92 million inputs. SW had the highest number of transactions (~541,000), while WW1 and WW2 under zkSNACKs produced around 35,000 each. Although SW's overall inflows were smaller compared to WW1, WW2 under zkSNACKs retained a significantly higher proportion of unspent output value. After mid-2022, WW2 absorbed significant activity from WW1. When all major coordinators for WW2 and SW were shut down between April and June 2024 after FBI actions, WW2's *kruw.io* eventually emerged as the dominant coordinator, now capturing over 99% of mixed liquidity with at least one coinjoin transaction every hour, each typically with more than 20, but up to hundreds ₿ on its input.

5.2 Study limitations

The main uncertain parameter of the analysis is the number of involved real separate entities (wallets) and their coin profiles. However, a large majority of the reported results does not depend on this (unknown) parameter.

The detection algorithm omits candidate coinjoin transactions for WW1 and WW2 with fewer than 15 inputs due to a significant increase in false positives if this threshold is further lowered. As a result, some coinjoin transactions are missing from the analysis, and a coordinator operating below this threshold may be omitted from our results; however, its impact on the analysis is small as a majority of such small transactions are present in collected ground-truth datasets (see Figure 6). SW is not impacted as these transactions are detected using a different mechanism.

The active participation in coinjoins as described in Section 3 is relatively expensive, both in capital expenses (if mixing behavior of larger denominations, e.g., above 1 ₿ is to be monitored) and operational expenses (at least mining fees and potentially also coordination fees need to be paid), preventing us from continuously performing client-side participation.

Due to the already extensive scope, we intentionally omit analysis of coinjoin designs other than those with a centralized but trust-minimized coordinator. The analysis of ones with decentralized trust-minimized coordinators (e.g., JoinMarket) or ones with a centralized trusted coordinator (e.g., ChipMixer) might benefit from some ideas from this paper (in particular, transaction-centric liquidity visualizations from Section 4.3.2), but requires deeper customization regarding transaction detection and interpretation of input and output liquidity meaning. For the same reason, we also omit discussion of coinjoin designs' resiliency against particular deanonymization attacks, which deserves research on its own.

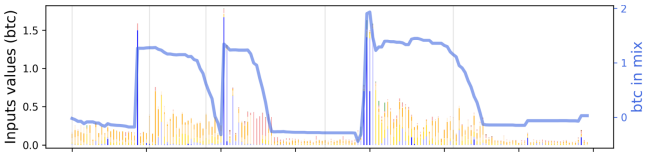

Figure 11: Visual detection of "fin"-like mixing patterns of interim liquidity caused by large wallet(s) under WW2 *opencoordinator* for a specific five days interval in March 2025. The pattern starts with fresh inflow, followed by gradual decrease as mixed coins are spent away from mix.

5.3 Privacy implications

Common intuition holds that more participants improve privacy and that large flows are harder to conceal. Our results show the expected scale of concurrent participants in a coinjoin and demonstrate that neither the use of a non-public coordinator nor substantial input flows can be effectively hidden.

Improved understanding of anonymity set: The estimation of actively participating wallets from Section 4.4 provides an insight into the expected number of other participants, the major contributing factor to the anonymity score of coins mixed. The estimate currently predicts more than 70 active participants for *kruw.io* coordinator on average, but significantly less for small coordinators, e.g., <10 for the *opencoordinator* coordinator, making it easier to mount a successful Sybil attack in such a case. Note that the wallet count estimation assumes all clients employ an unmodified standard CoinJoin wallet. This may overestimate the number of distinct wallets if modified clients register more inputs on average. Nonetheless, the assumption remains conservative regarding Sybil attacks, since such modifications would only amplify their potential effect.

Visibility of duration and post-mix actions for larger wallets: Even moderately sized mixed inputs are discernible for smaller coordinators, e.g., the "fin"-like patterns in *opencoordinator* around 1 ₿ visible at Figure 11, revealing the start and end of the wallet's mixing activity. Knowledge of the duration allows an analyst to attribute a significantly higher probability of ownership for outgoing outputs (MIX_LEAVE) to the large wallet in coinjoins executed during the descending part of the "fin"- like pattern. These patterns are also visible for large coordinators like *kruw.io*, but only for wallets with substantially larger inputs (e.g., two orders of magnitude more than for small coordinators) and are also less distinct due to the aggregate activity of a higher number of other users.

Non-public coordinator will not hide its presence: As long as the structure of its coinjoins resembles the generic coinjoin pattern with the same output denomination groups (especially when using the same coordination software), the transactions will be extracted from on-chain data, even if the coordinator is non-public. Such detection is especially effective when combined with active monitoring of all known coordinators, making the presence of non-public coordinator(s) quickly visible as shown on Figure 6.

6 Conclusions

This paper advances beyond prior work by introducing methods for detecting individual coinjoin coordinators, rather than merely identifying coinjoin transactions — a task of particular importance in the

post-zkSNACKs era, characterized by multiple competing coordinators. Our coordinator attribution algorithm retrospectively linked coinjoins to coordinators prior to public monitoring started in late June 2024, and uncovered a previously unknown non-trivial coordinator. We further show that the now-dominant *kruw.io* coordinator has already reached the pre-shutdown activity levels. In May 2025, the Whirlpool also re-emerged under the new anonymous Ashigaru coordinator, but yet to attract any significant liquidity.

We used distributions of outputs obtained from our client-side experiments to provide an estimator for the number of wallets active in a given coinjoin—a crucial anonymity parameter for the participating users. We further propose a novel transaction-centric liquidity analysis with sufficient granularity to isolate the activity of wallets mixing values significantly larger than the sum of all other active other participants active at the same time.

Future research directions may include several key areas: 1) Enhancing the wallet count predictor by incorporating additional features beyond the number of coinjoin outputs used in this study. The main challenge lies in the scarcity of ground-truth data on wallet coin distributions, anonymity targets, input–output correlations, and client code modifications. 2) Evaluating adversarial participant behavior, such as Sybil attacks involving numerous registered inputs or manipulations by coordinators, along with the corresponding game-theoretic implications. 3) Quantifying the anonymity loss of large wallets, which is currently observable visually, including if several large wallets participate concurrently. 4) Developing data-driven, actionable recommendations for developers regarding anonymity metric computations and appropriate default anonymity targets.

Ongoing analysis and monitoring of this ecosystem are valuable both for privacy-conscious users as well as law enforcement agencies. To support continued research, we provide open-source tools, regularly updated coinjoin datasets, and daily-updated visualizations at https://crocs-muni.github.io/coinjoin/.

Acknowledgments

We would like to thank anonymous reviewers and Lucas Ontivero for their comments, which truly helped improve the paper, and Stanislav Bobon for help with BlockSci verification results. The authors used generative AI-based tools to revise the text, improve flow, and correct any typos, grammatical errors, and awkward phrasing. Petr Svenda and Jiri Gavenda were supported by the EU project CHESS #101087529.

References

[1] Wabisator. https://wabisator.com/, visited on 2025-12-12.
[2] TDevD Adam Ficsor. ZeroLink: The Bitcoin Fungibility Framework, 2022. https://nopara73.medium.com/introducing-zerolink-the-bitcoin-fungibility-framework-dc5338086198, visited on 2025-12-12.
[3] Bitcoin.it. Bitcoin privacy wiki, 2019. https://en.bitcoin.it/Privacy, visited on 2025-12-12.
[4] BitLaundry. Bitlaundry - for all your bitcoin washing needs. https://web.archive.org/web/20160226223107/http://app.bitlaundry.com/, visited on 2025-12-12.
[5] Bybit. Bybit security incident, 2025. https://learn.bybit.com/this-week-in-bybit/bybit-security-incident-timeline/, visited on 2025-12-12.
[6] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.
[7] Adam Ficsor. Dumplings, 2020. https://github.com/nopara73/Dumplings, visited on 2025-12-12.
[8] Adam Ficsor, Yuval Kogman, Lucas Ontivero, and Istvan Andras Seres. Wabisabi: Centrally coordinated coinjoins with variable amounts. Cryptology ePrint Archive, Paper 2021/206, 2021. https://eprint.iacr.org/2021/206.
[9] Bitcoin Fog. Bitcoin fog clearnet portal. https://web.archive.org/web/20151108162913/http://bitcoinfog.com/, visited on 2025-12-12.
[10] Jiri Gavenda, Petr Svenda, Stanislav Bobon, and Vladimir Sedlacek. Analysis of input-output mappings in coinjoin transactions with arbitrary values. In Vincent Nicomette, Abdelmalek Benzekri, Nora Boulahia-Cuppens, and Jaideep Vaidya, editors, *Computer Security – ESORICS 2025*, pages 126–146, Cham, 2025. Springer Nature Switzerland.
[11] Roger Berger George Casella. Statistical inference, 2002.
[12] George Maxwell. Coinjoin: Bitcoin privacy for the real world, 2013. https://bitcointalk.org/index.php?topic=279249.0, visited on 2025-12-12.
[13] Sarah Meiklejohn, Marjori 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 *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, page 127–140, New York, NY, USA, 2013. Association for Computing Machinery.
[14] Malte Möser and Rainer Böhme. The price of anonymity: empirical evidence from a market for bitcoin anonymization. *J. Cybersecur.*, 3(2):127–135, 2017.
[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.
[16] Satoshi Nakamoto. Bitcoin core, 2018. https://github.com/bitcoin/bitcoin, visited on 2025-12-12.
[17] Hugo Schnoering and Michalis Vazirgiannis. Heuristics for detecting coinjoin transactions on the bitcoin blockchain. *arXiv preprint arXiv:2311.12491*, 2023.
[18] Rainer Stütz, Johann Stockinger, Pedro Moreno-Sanchez, Bernhard Haslhofer, and Matteo Maffei. Adoption and actual privacy of decentralized coinjoin implementations in bitcoin. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, AFT '22, page 254–267, New York, NY, USA, 2023. Association for Computing Machinery.
[19] Samourai team. Samourai wallet - whirlpool. https://web.archive.org/web/20240417214653/https://www.samouraiwallet.com/whirlpool, visited on 2025-12-12.
[20] Tin Tironsakkul, Manuel Maarek, Andrea Eross, and Mike Just. The unique dressing of transactions: Wasabi coinjoin transaction detection. In *Proceedings of the 2022 European Interdisciplinary Cybersecurity Conference*, EICC '22, page 21–28, New York, NY, USA, 2022. Association for Computing Machinery.
[21] Turbolay. Liquisabi. https://liquisabi.com/, visited on 2025-12-12.
[22] USAO. Sealed superseding indictment s2 24 cr.82, 2024. https://www.justice.gov/usao-sdny/pr/founders-and-ceo-cryptocurrency-mixing-service-arrested-and-charged-money-laundering, visited on 2025-12-12.
[23] Anton Wahrstätter, Alfred Taudes, and Davor Svetinovic. Reducing privacy of coinjoin transactions: Quantitative bitcoin network analysis. *IEEE Transactions on Dependable and Secure Computing*, 21(5), 2024.
[24] Sparrow Wallet. Sparrow bitcoin wallet. https://sparrowwallet.com/, visited on 2025-12-12.
[25] Wasabi Wallet. Wasabi wallet - bitcoin privacy wallet with coinjoin. https://wasabiwallet.io/, visited on 2025-12-12.
[26] Lei Wu, Yufeng Hu, Yajin Zhou, Haoyu Wang, Xiapu Luo, Zhi Wang, Fan Zhang, and Kui Ren. Towards understanding and demystifying bitcoin mixing services. In *Proceedings of the Web Conference 2021*, WWW '21, page 33–44, New York, NY, USA, 2021. Association for Computing Machinery.
[27] zkSNACKs. Wasabi wallet. https://github.com/WalletWasabi/WalletWasabi/tree/v1.98.4.0, visited on 2025-12-12.
[28] zkSNACKs. zksnacks is discontinuing its coinjoin coordination service 1st of june, 2nd may, 2024, 2024. https://web.archive.org/web/20240521005656/https://blog.wasabiwallet.io/zksnacks-is-discontinuing-its-coinjoin-coordination-service-1st-of-june/, visited on 2025-12-12.
[29] zkSNACKs Ltd. zkSNACKs - Unfairly private in the digital age. https://zksnacks.com/, visited on 2025-12-12.

A Ethics

Since our evaluation utilizes on-chain coinjoin data, user safety was a primary concern during our study. When designing our methodology, we have contacted the Wasabi team to discuss the safety precautions we put in place and shared early drafts of our work with developers of coinjoin protocols to provide responsible disclosure of our findings.

Our client-side participation naturally increased the anonymity set for all coinjoin users at the time of experiments, including potentially unlawful ones, but only mildly, given at least higher tens of other participants and a limited duration of our experiments.

The coordinator is a public trust-minimized service and attribution of coinjoin transactions it coordinates is not increasing or

decreasing anonymity of participants – the typical conservative assumption is that (fresh) transaction inputs owners are known before participation in coinjoin.

We do not attempt to deanonymize a human operator behind any coordination service; we only use it to infer the existence of such a separate entity (e.g., an unknown coordinator). The related dataset contains only data made publicly available by a given coordinator to all connected wallets during their regular operation; the released dataset provides only a historical archive of the data.

We believe that the only non-public and truly sensitive pieces of information in our research were client-side logs collected during client-side experiments, as all other data is publicly available on blockchain, in public databases of third-party monitoring services, or was obtainable via the coordinator's public API. We do not release these client-side logs; instead, we provide only aggregated statistics here.

A.1 Adherence to coordinator's usage policy

Coordinators typically have no usage policy specified, as they are anonymous with only bare usage instructions. The exceptions are the WasabiWallet project (WW1 and WW2 during zkSNACKs coordinator) and its fork (WW2 GingerWallet). Existing policies permit monitoring provided it does not adversely affect the service experienced by other users – which is not our case due to only occasional standard API requests. The source code is licensed under MIT, GPL, or aGPL.

The original zkSNACKs coordinator for WW1 and WW2 (files WalletWasabi/Legal/Assets/LegalDocumentsWw1.txt and *Ww2.txt and also available via GetLegalDocuments() API call) does not forbid monitoring as long as it does not negatively impact service for other users. WalletWasabi repository (post-zkSNACKs) removed recently (26th April 2025) legal documents by commit b98d3f0[13] altogether.

The GingerWallet code (fork of zkSNACKs's WalletWasabi code, LegalDocumentsGingerWallet.txt[14] introduced only minor changes to the original zkSNACKs license, related only to the renaming of a development team and operator company.

Samourai Whirlpool coordinator (SW) had no EULA or TOS published, only a code license (GPL) till the FBI seizure. An older mirror for the code[15] is available. Recently launched Ashigaru Whirlpool (AW), which is a continuation of Whirlpool, has no specific EULA or TOS at its clearnet[16] page or Tor onion site[17].

B Liquidity flows discussion

As of 10th November 2025, WW1 and WW2 under zkSNACKs coordinator have about 2.5% and 4.4% of non-remixed coinjoin outputs still unspent, with 3,456 ₿ and 4,880 ₿, respectively, after more than a year of inactivity. SW has significantly more unspent outputs (average 9.5% over all its pools, but 18.9% for 50M pool), in total value of 7,315 ₿.

Figure 9 provides insights into the liquidity dynamics of all three studied designs. WW1 cumulative liquidity grows until the beginning of 2022, after which liquidity outflows become higher than fresh inflows, including remixing previously mixed outputs before withdrawing them from a pool. The fresh liquidity decline of WW1 corresponds to the move into a newer WW2 pool by users simply upgrading to a newer Wasabi Wallet 2.x version in 2022. Occasional changing liquidity peaks (e.g., January 2024 for WW2) are caused by a large inflow of fresh funds (presumably one or a few very large wallets), mixed and sent out shortly after.

WW2 pool cumulative liquidity quickly rose to around 4,000 ₿ in December 2022, after which it slightly decreased to around 3,000 ₿ and started again slowly rising back to around 4,000 ₿ over the whole year 2023. The sharp increase came during April and especially May 2024, after the plan to shut down the WW2-zkSNACKs coordinator was announced. The users likely rushed to mix while still operational, which resulted in a sharp increase in liquidity inflows as well as the value of unspent outputs above 7,000 ₿. The post-zkSNACKs coordinators were slower to accumulate unmoved outputs, with a significant rise only from the start of 2025, especially under *kruw.io* coordinator.

SW displays very different dynamics from WW1 and WW2. SW unspent coins grow more continuously and exhibit less volatility. The changing liquidity plot line is smoother due to larger inputs being first split into a particular pool denomination via *TX*0 before entering any coinjoin transaction, spreading the (larger) input over multiple transactions in time. Finally, the typically fresh daily inflows are also significantly smaller than for WW1 and WW2. Despite that, SW had the highest fraction of mixed, but unmoved outputs, showing a different prevalent usage pattern of "mix then hold". Only during April 2025, the unspent coins from all WW2 coordinators combined surpassed it.

As seen from Figure 9, the remix rate (input liquidity repeatedly mixed to achieve higher anonymity) was mostly around 62.3% for SW (3 out of 5 inputs, later rising up to 75% for 6 out of 8 inputs), on average 72.8% for WW1, but significantly higher at 94.6%+ for WW2. The different remix rates are directly influenced by the given coinjoin design as well as the structure of fees.

Finally, there are liquidity flows between post-ZKSNACKs coordinators under WW2 coinjoin design, as users are changing the coordinator in their wallet configuration and mixing already mixed funds under a new one. On the Figure 12, flows between significant WW2 coordinators are highlighted. As expected from the fact that *kruw.io* is currently the dominant coordinator, most of the outflows from *zksnacks* are towards it, but with non-trivial flows also from *opencoordinator* and *wasabicoordinator*. The lower part of Figure 12 shows the situation only for mixed outputs created after 1st January 2025, when only four coordinators remained operational. While *kruw.io* still receives most of the previously mixed flows from other coordinators (76.4 ₿ total) with *opencoordinator* being the largest contributor (69.7 ₿), a significant portion was also sent from the *kruw.io* to the *opencoordinator* (42.0 ₿). Similarly, although on a smaller scale, *coinjoin_nl* received 5.26 ₿ from *kruw.io* and sent 4.87 ₿ back to it. We hypothesize that some users regularly use multiple coordinators with the goal of achieving better anonymity.

[13] https://github.com/WalletWasabi/WalletWasabi/commit/b98d3f0972cfd4168fc66e1b0deb5719b8552410
[14] https://github.com/GingerPrivacy/GingerWallet/blob/master/WalletWasabi/Legal/Assets/LegalDocumentsGingerWallet.txt
[15] https://github.com/Samourai-Wallet/Whirlpool
[16] https://ashigaru.rs
[17] http://ashicodepbnpvslzsl2bz7l2pwrjvajgumgac423pp3y2deprbnzz7id.onion/Ashigaru/

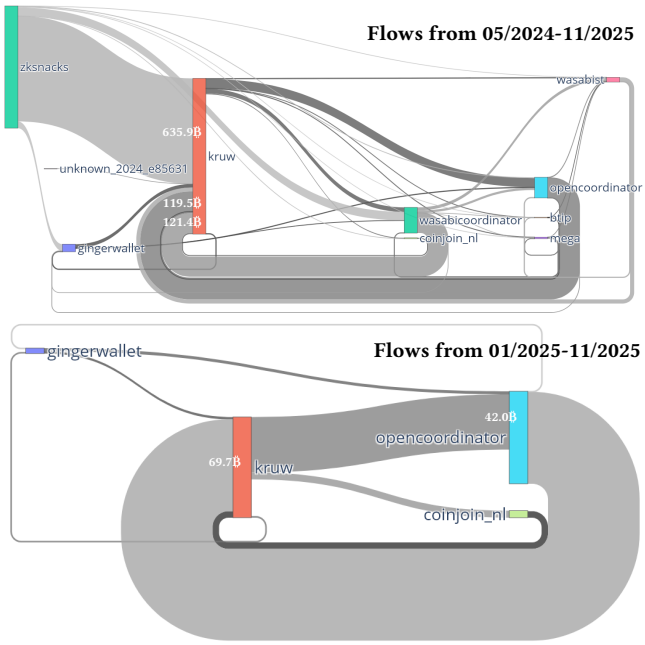

Figure 12: Flows of previously mixed liquidity between WW2 coordinators for the whole studied period since May 2024 (upper picture) and filtered only for new outputs created from January 2025 (lower picture). Self flows (remixes under the same coordinator), very small coordinators and very small flows to zkSNACKs are omitted for picture clarity. Input flows to *kruw.io* and *opencoordinator* are annotated with values in ₿ for scale.

C Specifics of coinjoin protocols mixing process

In this section, we describe an example mixing process under different designs step by step when starting with 1 ₿. In the description, we assume that the wallet has initially just one coin. However, if there are already other existing coins in the wallet, the situation will be very similar, only with previously existing coins also included in the mixing. All implementations require active online participation of the wallet controlling the coins.

C.1 Wasabi Wallet 2.x

When a user wants to mix 1 ₿ with Wasabi Wallet 2.x the process goes as follows:

(1) The user sends 1 ₿ to his Wasabi Wallet.
(2) A user selects desired coordinator (e.g., *kruw.io*) by setting coordinator's API endpoint. The coordinator can be changed by the user at any time.
(3) The wallet automatically registers one or more coins (randomized, but considering the given coin's current *anonscore* to prefer coins without the desired target *anonscore* threshold) as an input for the next coinjoin.
(4) When the output registration period starts, the wallet automatically selects a suitable split of output values and registers outputs of these values (together equaling to 1 ₿ minus coordination and mining fees).
(5) When the output registration period ends, the wallet requests information about transaction inputs and outputs, constructs deterministically and locally the transaction, checks if all inputs and outputs of this wallet are included, and signs its input(s). The signature(s) is sent back to the coordinator.
(6) Transaction is broadcast to the Bitcoin network by clients or the coordinator (was changing over time).
(7) The wallet computes the *anonscore* of all its newly created coins. If all of them have sufficiently high *anonscore*, the process ends. Otherwise, the wallet continues from Step 2, potentially registering multiple inputs.

The desired level of anonymity is set by the user via the target *anonscore* parameter, which is computed on a per-coin basis. The mixing time for a default *anonscore* value is typically in hours.

C.2 Wasabi Wallet 1.x

The mixing process of 1 ₿ with Wasabi Wallet 1.x is as follows:

(1) The user sends 1 ₿ to his Wasabi Wallet, becoming a coin in the wallet.
(2) The user chooses one (or more) coin(s) to mix and sets a desired *anonset* anonymity target.
(3) The coordinator was (as no WW1 coordinator is operational now) fixed by default in wallet software, pointing to the zSNACKs coordinator.
(4) The wallet registers the input into the next coinjoin.
(5) The wallet registers output(s) (in version 1.0, only one output of 0.1 ₿, in 1.1, the wallet can register one output for each allowed denomination, e.g., 0.1, 0.2, 0.4,...).
(6) The wallet computes *anonset* of all its outputs.
(7) Transaction is broadcast to the Bitcoin network by each participating wallet.
(8) If some outputs do not meet the desired anonset yet and have sufficient value to obtain at least an output of 0.1 ₿, then the wallet registers them into the next coinjoin and continues as described in Steps 3-6. Otherwise, mixing stops.

The desired level of anonymity is set by the user via the target *anon set* parameter, which is computed on a per-coin basis. The mixing time for a default *anonscore* value typically takes hours.

C.3 Samourai/Ashigaru Whirlpool

When mixing 1 ₿ with the Samurai/Ashigaru Whirlpool, the following steps happen:

(1) The user sends 1 ₿ to his Whirlpool client wallet, becoming a coin in the wallet.
(2) The user manually chooses a suitable mix pool based on which denominations are desired (e.g., 0.05 ₿ pool). The Whirlpool coordinator endpoint is fixed by default in the wallet software.
(3) The user initiates creation of special transaction *TX0*, in which pays a coordination fee and obtains multiple outputs of a standard denomination (e.g., 19 outputs of slightly higher than 0.05 ₿; with small surplus used to pay for mining fee in subsequent coinjoin) and some change output (not mixed further).

(4) The wallet registers the *TX0*'s outputs of standard denominations as inputs to the queue for future coinjoin rounds. For each coinjoin, registering just one input. The coordinator chooses randomly from additional registered inputs in the queue (registered by other wallets) are selected for the round.
(5) For each round, where the wallet's input is included, the wallet provides an output address.
(6) For the first coinjoin of each *TX0*'s output, the coinjoin is typically very quick, as this output pays for the mining fee by its (small) surplus value over the pool's size. The subsequent remixes (where someone else is paying the mining fee) take typically significantly longer to execute.
(7) Wallet checks the proposed transaction and, if it is correct, signs it.
(8) Transaction is broadcast to the Bitcoin network by the coordinator.
(9) The wallet automatically registers mixed coins for other coinjoin rounds (if coins are selected, Steps 4-6 are followed).
(10) Mixing ends when the user decides to send the funds out of the Wallet. Any leftover from send (change output) staying in the Wallet is *not* automatically mixed further.

The desired level of anonymity is controlled by a user observing the number of remixes for each coin performed so far. The mixing time for a default number of remixes (=5) typically takes at least days or more.

D Evaluation of Wasabi 2.x coordinator attribution algorithm

In this section, we provide a rationale behind the selection of specific threshold values and the inner working of the coordinator attribution algorithm and evaluate its sensitivity to increasingly larger fractions of transactions to be attributed.

D.1 Same-coordinator remix ratio and resulting attribution threshold

As the coordinator attribution algorithm utilizes a threshold for the fraction of inputs (and outputs) assumed to come from transactions previously created under the same coordinator, we first established a suitable threshold based on the inter-coordinator remix rate using the distribution of real values observed from the ground-truth dataset, as shown in Figure 13. The established threshold of 40% (used during subsequent analyses) separates the first two coordinators with a large margin most of the time, as can be seen in Figure 14. The separation is significant especially after the brief period of June-August 2024, during which new coordinators were started and seeded with already mixed funds from other existing coordinators (typically zkSNACKs).

D.2 Sensitivity to unattributed transactions

While the collected ground-truth mapping datasets attribute a large majority of extracted coinjoin transactions based on the coordinator's API monitoring, 3065 transactions (6.88%) were not unattributed due to gaps in monitoring, mostly till November 2024. We therefore analyze the sensitivity of the coordinator attribution

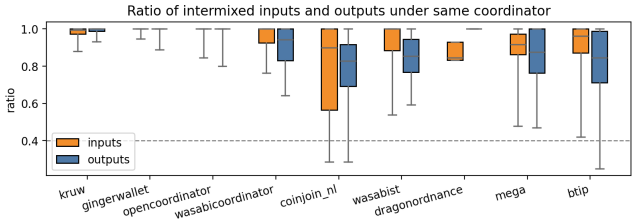

Figure 13: Distribution of ratios of inputs and outputs coming from coinjoin transactions coordinated under the same coordinator observed in the ground-truth dataset. The boxplot's whiskers extend to the 5th and 95th percentiles of the data. Except for the *coinjoin_nl* coordinator, all inputs are above 40% threshold.

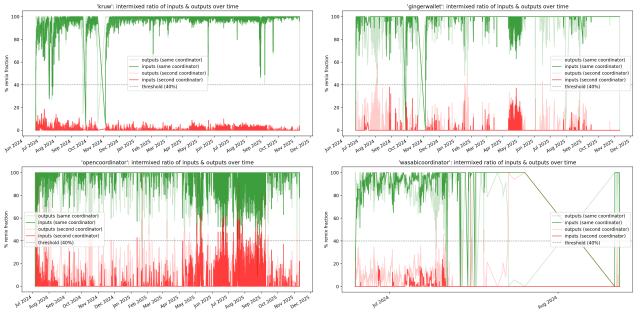

Figure 14: Ratios of inputs (dark) and outputs (light) originating from coinjoin transactions attributed to the same coordinator and the second most common one (which may vary over time), computed from the ground-truth dataset for four representative coordinators. Inputs from the same coordinator are green, inputs from the second most common coordinator are red (the exact coordinator responsible may change over time). Occasional significant drops in the same-coordinator ratio result from gaps in the ground-truth data, where missing transactions cause inputs to appear—temporarily—as originating from other unattributed coordinators.

algorithm (see Section 4.2.4) to an increasing fraction of initially unattributed transactions.

Using the aggregated ground-truth dataset, we moved an increasing fraction of attributed transactions and placed these into the unattributed set. The selection of transactions for removal is performed for a specific coordinator either 1) randomly (corresponds to occasional omission of transactions during monitoring) or 2) continuously from the end (corresponds to omission of all coordinators' transactions after a certain moment in time). After the removal, the coordinator attribution algorithm is rerun, with its output compared against the results expected from the aggregated ground-truth dataset, counting false positives and false negatives. Note that exhaustive evaluation over all potentially variable parameters is impractical due to too many combinations (nine known

coordinators, hundreds to thousands of transactions each, transaction removal approach, time period, etc.).

We performed the following three experiments:

- **Random txs, single coordinator**: drop of transaction attribution of 0-100% for each coordinator separately, with results averaged over 10 executions, visualized at Figure 15.
- **Random txs, any coordinator**: drop of 0-100% from all transactions under any coordinator, averaged over 10 executions, visualized at Figure 16.
- **Tail txs, single coordinator**: drop of last 0-100% transactions for each coordinator separately (deterministic, no averaging needed), visualized at Figure 17.

The results are shown on Figure 15, demonstrating that the algorithm is notably insensitive to transactions with missing attribution until around 80% of transactions are randomly dropped (false positives). The algorithm is attributing transactions properly when trailing transactions of a single coordinator are dropped for almost all coordinators, with the exception of *opencoordinator*, which is significantly misattributed after removal of 33% of trailing transactions as seen on Figure 17.

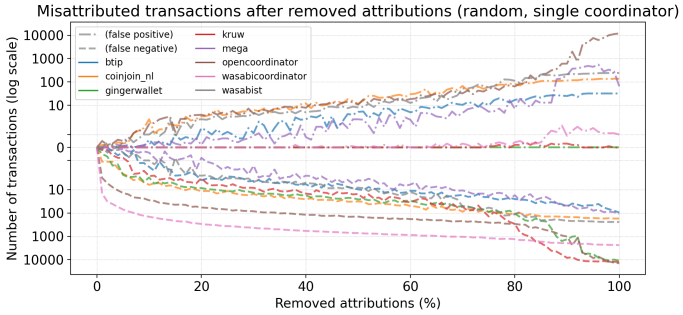

Figure 15: Impact of removal of increasing fraction of attributed transactions (txs) on coordinator attribution algorithm, measured in number of misattributed txs for: random txs of a single coordinator (top graph), random txs of any coordinator (middle graph), and tail txs (lowest graph). The impact of the removal of random txs of a single coordinator is generally small, with all but *opencoordinator* coordinators staying below 200 misattributed txs. Until the removal of up to 80% randomly selected txs from any coordinator provides only a small linearly increasing number of misattributed txs, after which it quickly rises. Finally, dropping all trailing txs of a single coordinator results in an excessive number of misattributed transactions after 33% for *opencoordinator*, 91% for *mega*, and 96% for *wasabist* coordinators, respectively; the rest of the coordinators see small to negligible numbers of misattributed txs. Note the logarithmic scale of the y-axis.

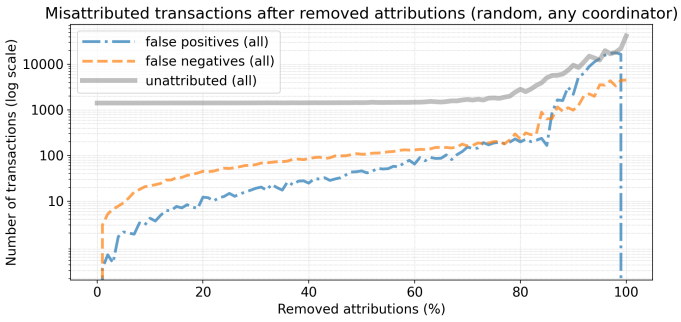

Figure 16: Impact of removal of an increasing fraction of random transactions from any of the known coordinators. Until the removal of up to 80% randomly selected transactions, only a small and linearly increasing number of misattributed txs is observable. Note the logarithmic scale of the y-axis.

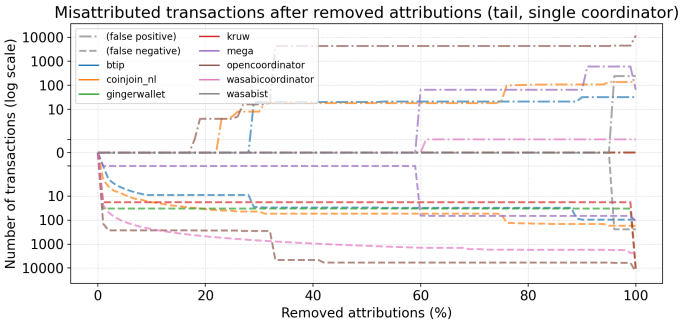

Figure 17: Impact of removal of a fraction of tail transactions for a single coordinator on the coordinator attribution algorithm. All but *opencoordinator* coordinator stay below 100 incorrectly attributed transactions (false positives) until around 76% of trailing transactions are removed. The number of transactions not attributed to their real coordinator (false negatives) is significant only for *opencoordinator* and *wasabicoordinator* due to generally low internal remix ratio (below our threshold 0.4). Note the logarithmic scale of the y-axis.

E Newly discovered coordinators

The active monitoring of known coordinators, combined with all coinjoin-like transactions observed on the blockchain allowed us to identify two previously unknown coordinators in the set difference between these two sources. As seen on Figure 6, the primary target period is May to November 2024 (red region), after which very few transactions remained unattributed. Using the coordinator attribution algorithm, we found two distinct clusters of transactions, previously unknown coordinator(s) named *unknown_2024_e85631* (shown on Figure 18) and *unknown_2024_28ce7b*.

unknown_2024_e85631 was operational from early May 2024 (the earliest post-zkSNACKs coordinator detected) to 11th November 2024, mixing around 38 ₿ over 653 transactions, while *unknown_2024_28ce7b* mixed only about 9 ₿ over 55 transactions within a short period of 14-25th November 2024. Both clusters of transactions for these coordinators have almost always 100% remixes from their own transactions, with practically no mix inputs coming from other coordinators, consistent with a private coordinator scenario with clients not switching between several

coordinators. Given the same distinct intermix ratio and the second one starting only 3 days after the previous one stopped, we can hypothesize that both clusters are coordinated under the same underlying non-public entity.

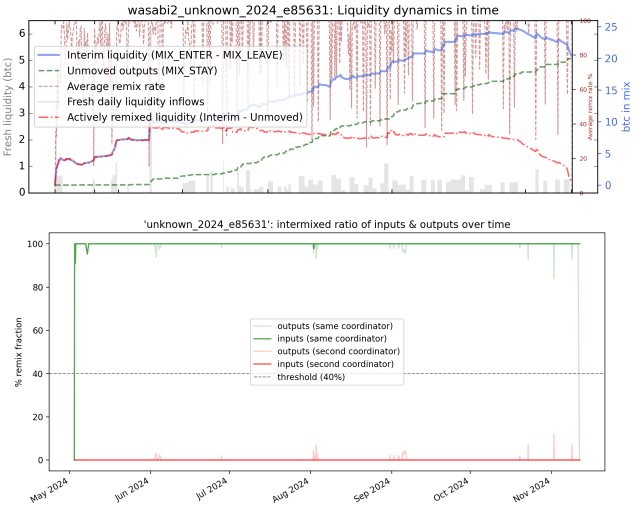

Figure 18: Mixed liquidity and intermix ratio for previously unknown coordinator *unknown_2024_e85631* (653 txs), operational from early May 2024 to November 2024, mixing around 38 ₿. Remixed inputs and outputs are almost entirely (nearly 100%) processed under the same coordinator, consistent with a scenario in which clients mix exclusively through a non-public coordinator without switching to other coordinators.

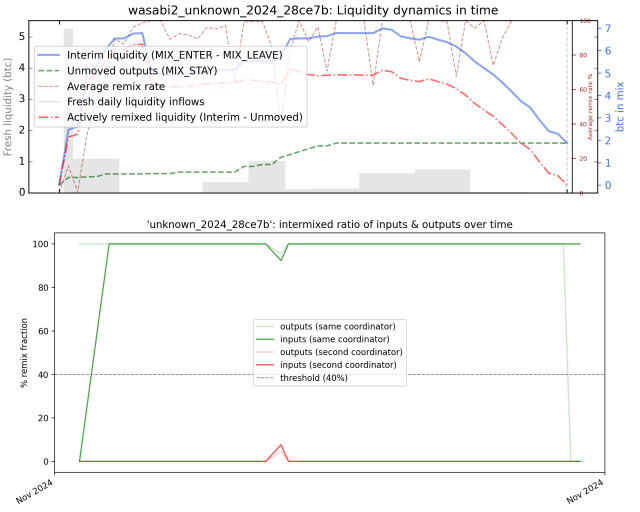

Figure 19: Mixed liquidity and intermix ratio for previously unknown coordinator *unknown_2024_28ce7b*, mixing around 9 ₿ over 55 txs in November 2024.

F Wallet prediction error bounds for WW2

To estimate the number of unique participating wallets in a coinjoin transaction, we model the number of inputs or outputs registered per wallet as an independent and identically distributed (i.i.d.) random variable inferred from client-side measurements. Let N denote the total number of inputs or outputs observed in a transaction. If the mean number of coins registered by a wallet is μ, the plug-in estimator for the number of wallets is

$$\hat{W} = \frac{N}{\hat{\mu}}$$

where $\hat{\mu}$ is the empirical mean number of coins per wallet obtained from our client-side measurements. This estimator is a smooth function of $\hat{\mu}$ and therefore admits a first-order variance approximation via the delta method. Writing $g(\mu) = N/\mu$, we have $g'(\mu) = -N/\mu^2$, which quantifies how uncertainty in $\hat{\mu}$ propagates into uncertainty in $\hat{W}$. The asymptotic variance is thus approximated by

$$\mathrm{Var}(\hat{W}) \approx \left(\frac{N}{\hat{\mu}^2}\right)^2 \mathrm{Var}(\hat{\mu}) \tag{1}$$

where $\mathrm{Var}(\hat{\mu})$ is obtained from the empirical per-wallet distribution.

We report 95% Wald confidence intervals (CI) using this variance approximation. This approach is appropriate under the assumption that all participating wallets are unmodified and that their coin-registration behaviour follows the same underlying distribution. Adversarial deviations (e.g., such as Sybil participants registering arbitrarily many inputs or outputs) break these assumptions and degrade privacy by inflating $\hat{W}$ while shrinking the true anonymity set. Consequently, the confidence intervals reported here characterise the privacy achievable under honest participation and should be interpreted as upper bounds on the effective anonymity that the system can provide.

Figure 20 presents the resulting 95% CIs for four cases, corresponding to two independent experiments (as25 for zkSNACKs and as38 for kruw.io) and two sources of observational data (inputs or outputs). Note that the input- and output-based predictions are not expected to align for the same value of N, as real coinjoin transactions typically contain more outputs than inputs.

G Mining fee and hidden coordinator tips

Figure 22 shows the distribution of mining fees, fairly proportional to the number of coins registered by our wallet. As *as38* was executed during a period with lower on-chain fees (typically 2-3 sats/vB), its distribution is also naturally lower than for *as25*. Additionally, the surplus fee (which we call the hidden coordination tip) is computed as the difference between our inputs and outputs excluding the fair mining fee. The hidden coordination tip is paid when a wallet cannot split the sum of inputs into chosen output denominations and leaves a small leftover amount, which is then collected (in sum over all wallets) by the coordinator. The hidden tip is typically less than 100 sats[18] per wallet, but in some cases it can be significantly higher (up to 8472 sats observed), especially with increased minimum registrable amount for standard output denomination of the *kruw.io* coordinator (from 5000 to 9999 sats)—providing income for the coordinator even when the mandatory

[18] 1 ₿ = 100,000,000 sats

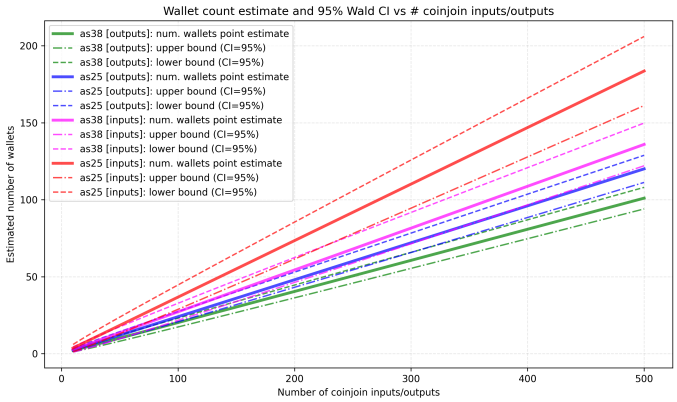

Figure 20: Error bounds (95% CI) for wallet predictions based on the observed distribution of inputs and outputs in client-side experiments. Differences between input- and output-based estimates for the same N are expected due to the typical coinjoin transaction structure (see text above) with a median number of inputs to outputs being 230 to 272 and a maximum of 498 to 652 for *kruw.io* (experiment *as38*). Similarly, for *zkSNACKs* (experiment *as25*), it is 205 to 226 (median) and 400 to 508 (maximum).

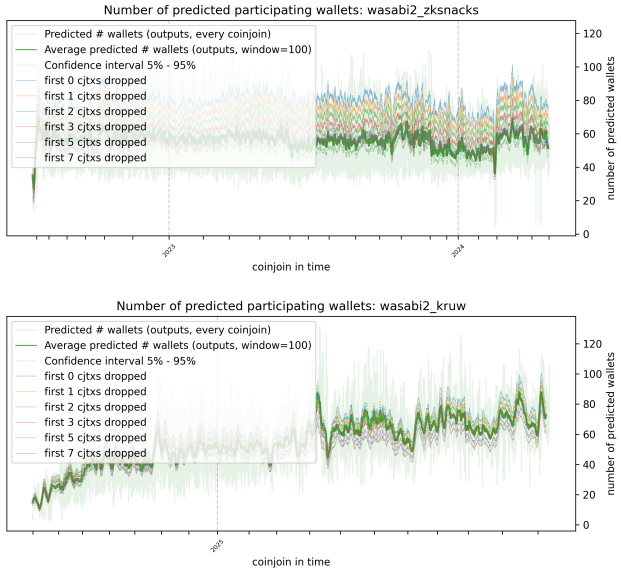

Figure 21: Impact of omitting X first coinjoins from real client-side experiments (until the wallet is populated with enough coins) to match outputs-based predictions (bolt green line). *zksnacks* required omitting the first five coinjoins in comparison to only three for *kruw.io*, suggesting that significantly more wallets under *kruw.io* have a small number of mixed coins.

coordination fee was 0%. Rarely, the observed hidden coordination tip value was even negative (max. -96 sats observed), in which case a wallet pays slightly less than its fair mining fee share.

Any input-to-output deanonymization algorithm must account for these distributions, since the inputs of a targeted wallet are reduced by these characteristics.

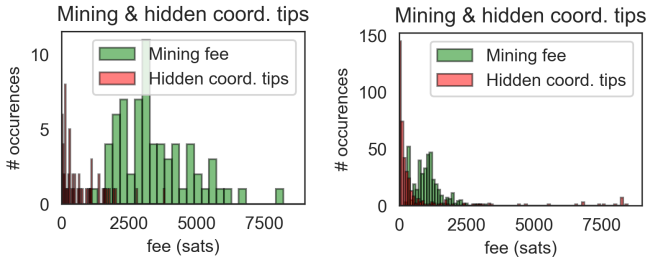

Figure 22: Distribution of mining fees and hidden coordination tips observed for *zkSNACKs* (left) and *kruw.io* (right) experiments. Each dataset is represented using 100 bins.

H Whirlpool waiting queue length estimation

Table 2 provides an estimation of the number of UTXOs waiting for remix via average queue length (*Est.q.len.*) for different pool sizes of Samourai Whirlpool. Based on our wallet's observed participation characteristics (*Our participation* column) and all Whirlpool coinjoins executed during the participation interval extracted from on-chain data (*CJs in period* column), the waiting queue length can be estimated. Note that the waiting queue length is not identical to the potential anonymity set, since a single wallet may register multiple coins; it is only proportional to it. In Whirlpool, a wallet typically participates in multiple consecutive coinjoins (a session) for each coin, and the anonymity set corresponds to the *distinct* wallets providing the other inputs in these coinjoins. While a single coinjoin enforces input selection from different wallets, this does not necessarily hold across the entire session. Assuming queue entries are not heavily concentrated in just a few wallets, longer queues shall yield larger anonymity sets.

Table 2: Estimation of waiting queue length based on our Whirlpool client-side experiments for different pool sizes.

Coinjoin pool (date)	CJs in period (#/free inputs)	Our participation (coins/mixes/rate)	Est. q. len.
100k (2023-08-02—12)	2308 / 11797	**2 / 10 / 0.08%**	**1180**
100k (2023-08-14—09-13)	4938 / 20628	**2 / 35 / 0.17%**	**589**
1M (2021-11-04—11)	2342 / 6733	**10 / 41 / 0.61%**	**164**
1M (2021-11-12—21)	2062 / 5798	**10 / 30 / 0.52%**	**193**
1M (2021-11-25—12-03)	2144 / 5542	**12 / 38 / 0.69%**	**146**
1M (2021-12-05—24)	2205 / 6458	**4 / 19 / 0.29%**	**340**
1M (2023-08-14—09-13)	2702 / 11375	**12 / 32 / 0.28%**	**355**
5M (2023-07-17—29)	649 / 3436	**11 / 20 / 0.58%**	**172**
5M (2023-08-14—09-13)	1568 / 6771	**4 / 24 / 0.35%**	**282**

I Changes to Dumplings processing software

The original Dumpling code repository has the last changes from the end of the year 2023 (commit 36f28f2). To accommodate for changes in coinjoin transactions for post-zkSNACKs coordinators, we performed the following changes:

- Decrease of the minimum number of inputs for WW2 from 50 to 15.
- Scanning started from block 336861 (2015-01-01) to capture early JoinMarket transactions.
- Addition of 2.5M and 25M sats denominations to detect Whirlpool pools under the Ashigaru coordinator.
- Added detection of Taproot lock scripts and Replace-By-Fee transaction flag (BIP-125) and serialization to output files.

The changes are available in our Dumplings repository fork (https://github.com/crocs-muni/Dumplings) as the upstream repository is not maintained anymore.

J Data collected from client-side software

We used *unmodified* client software wallets to participate in WW1, WW2 and SW coinjoins to prevent any unintentional modifications to the client's behavior. All implementations produce some level of logging into log files, in addition to the information displayed by the wallet graphical interface. However, sensitive information like identification of wallet coins and their intermediate status is not written (intentionally) in case the log file would be exposed. Additionally, WW2 provides detailed information upon a RPC request. We collect the following files from a client wallet working directory and from RPC requests:

- WW1: Manual extraction of transactions, coins and their anonset value from Wallet Wasabi's graphical interface.
- WW2: Logs.txt (runtime client logs), PrisonedCoins.json (coins temporarily banned from participation), coins.json (wallet coins and their status obtained using `listcoins` RPC method), history.json (transactions in which the wallet participated, obtained using `gethistory` RPC method), daily screenshot of wallet UI with unspent coins and their anonscore shown (for manual cross-verification against coins.json).
- SW: Manual extraction of transactions, coins and their number of remixes from Sparrow Wallet's graphical interface.

However, these collected raw logs and other data are *not* made public as discussed in the Ethics section.

K Transaction-centric liquidity visualization pseudo-algorithms

Pseudocode of computation of transaction-centric, cumulative pool liquidity statistics used for visualization as described in Section 4.3.2. The input arrays are created from all sorted coinjoins after establishing the type of inputs and outputs as shown on Figure 2: MIX_ENTER, MIX_LEAVE, MIX_STAY, MIX_REMIX, MIX_REMIX_FRIENDS, MIX_REMIX_FRIENDS_WW1. The full working code is available in `cj_visualize.py::plot_mix_liquidity()`.

Input:
Number of coinjoins N;
Arrays $mix\_enter[i]$, $mix\_remixfriend[i]$, $mix\_remixfriend\_ww1[i]$, $mix\_leave[i]$, $mix\_stay[i]$ for $i = 0, \ldots, N-1$;
Array $mix\_leave\_time\_before[i]$ for $i = 0, \ldots, N-1$ (fast-leaving part of $mix\_leave[i]$, default 30 days);
Output:
Array $interim\_liquidity[i]$ for $i = 0, \ldots, N-1$.
Array $interim\_liquidity\_time[i]$ for $i = 0, \ldots, N-1$.
Array $stay\_liquidity[i]$ for $i = 0, \ldots, N-1$.
Array $remix\_liquidity[i]$ for $i = 0, \ldots, N-1$.

$curr\_liquidity \leftarrow 0$
$curr\_liquidity\_time \leftarrow 0$
$curr\_stay\_liquidity \leftarrow 0$
$curr\_remix\_liquidity \leftarrow 0$
for $i \leftarrow 0$ **to** $N-1$ **do**
  `// Interim pool's liquidity`
  base_liquidity_step $\leftarrow mix\_enter[i] + mix\_remixfriend[i] + mix\_remixfriend\_ww1[i] - mix\_leave[i]$
  $curr\_liquidity \leftarrow curr\_liquidity +$ **base_liquidity_step**
  $interim\_liquidity[i] \leftarrow curr\_liquidity$
  `// Interim pool's liquidity with time cutoff`
  $curr\_liquidity\_time \leftarrow curr\_liquidity\_time +$ **base_liquidity_step** $+ mix\_leave[i] - mix\_leave\_time\_before[i]$
  $interim\_liquidity\_time[i] \leftarrow curr\_liquidity\_time$
  `// Unmoved outputs`
  $curr\_stay\_liquidity + mix\_stay[i]$
  $stay\_liquidity[i] \leftarrow curr\_stay\_liquidity$
  `// Actively remixed liquidity`
  $remix\_liquidity\_step \leftarrow$ **base_liquidity_step** $- stay\_liquidity[i]$
  $curr\_remix\_liquidity \leftarrow curr\_remix\_liquidity + remix\_liquidity\_step$
  $remix\_liquidity[i] \leftarrow curr\_remix\_liquidity$
end

Algorithm 1: Pseudocode of computation of transaction-centric, cumulative pool liquidity statistics.

L All coinjoins statistics

Coinjoin pool (coordinator name)	Operating period (from – to)	Total cjtxs (#)	Fresh inputs (#/value)	Remix rate (average)	Unmoved UTXOs (#, value)	Num. of inputs (min/avg/max)
Wasabi 1.x (zkSNACKs)	2018-07-19 – 2023-05-19	33594	336k / 227231.2 ₿	72.8%	2.5%, 3456.0 ₿	6 / 74.3 / 186
Wasabi 2.x (zkSNACKs)	2022-06-18 – 2024-06-02	35170	273k / 70007.3 ₿	94.6%	4.4%, 4880.6 ₿	50* / 214.9 / 400
Wasabi 2.x (post-zkSNACKs)	2024-05-02 – running	42717	135k / 46958.3 ₿	87.6%	11.5%, 3501.0 ₿	<15 / 96.7 / 498
Wasabi 2.x (kruw.io)	2024-05-07 – running	13952	123k / 46309.7 ₿	87.5%	11.1%, 3388.3 ₿	<15 / 227.5 / 498
Wasabi 2.x (opencoord.)	2024-07-06 – running	14295	6524 / 257.9 ₿	89.3%	14.2%, 21.2 ₿	<15 / 32.5 / 100
Wasabi 2.x (gingerwallet)	2024-06-02 – running	11971	2467 / 187.4 ₿	95.1%	14.4%, 60.1 ₿	<15 / 33.6 / 81
Wasabi 2.x (wasabicoord.)	2024-06-09 – 2024-08-12	1187	1523 / 135.3 ₿	87.3%	3.0%, 7.3 ₿	<15 / 35.1 / 102
Wasabi 2.x (unknown_2024_e85631)	2024-05-02 – 2024-11-11	653	260 / 37.9 ₿	82.9%	14.0%, 19.9 ₿	<15 / 42.6 / 87
Wasabi 2.x (unknown_2024_28ce7b)	2024-11-14 – 2024-11-25	55	73 / 9.0 ₿	71.3%	8.6%, 1.9 ₿	<15 / 37.6 / 65
Wasabi 2.x (coinjoin.nl)	2024-12-04 – running	144	73 / 4.0 ₿	74.2%	29.0%, 2.1 ₿	21 / 26.4 / 42
Whirlpool all (Samourai)	2019-04-17 – 2024-05-01	541131	1.13M / 47309.0 ₿	62.3%	9.5%, 7315.0 ₿	5 / 5.4 / 8*
Whirlpool 5M	2019-04-17 – 2024-04-25	112143	233k / 11705.5 ₿	61.4%	9.4%, 1102.5 ₿	5 / 5.4 / 8*
Whirlpool 1M	2019-05-23 – 2024-04-24	205648	434k / 4359.8 ₿	60.0%	7.9%, 341.1 ₿	5 / 5.3 / 8*
Whirlpool 50M	2019-08-02 – 2024-04-25	30327	61k / 30826.9 ₿	63.0%	18.9%, 5832.5 ₿	5 / 5.5 / 8
Whirlpool 100k	2021-03-05 – 2024-05-01	193013	404k / 416.7 ₿	60.7%	9.6%, 38.9 ₿	5 / 5.4 / 8*
Whirlpool all (Ashigaru)	2025-05-31 – running	287	611 / 42.6 ₿	57.4%	72.7%, 41.0 ₿	5 / 5.0 / 5
Whirlpool 2.5M	2025-05-31 – running	218	445 / 11.1 ₿	59.2%	70.3%, 7.8 ₿	5 / 5.0 / 5
Whirlpool 25M	2025-06-06 – running	69	166 / 41.5 ₿	51.9%	80.1%, 33.2 ₿	5 / 5.0 / 5

Table 3: Summary of the operational parameters for studied coinjoin pools until 10th November 2025. Unmoved UTXOs are computed from non-remixed outputs. Fresh inputs value excludes part of the input value if followed by non-standard outflow (WW1 only) and "friends-do-not-pay" (inputs to WW2 coming from WW1 or sent between two WW2 users) liquidity. Values marked * are with some, but negligible exceptions. Operating period is based on the time when the last coinjoin transaction of a given coordinator was mined—the coordinator stopped coordination shortly before that. Note that some "post-"zkSNACKs coordinators were already operational while zkSNACKs coordinator was still running (but scheduled for shutdown). Coincidentally, the earliest detected non-zkSNACKs transaction is from an unknown coordinator *unknown_2024_e85631*.

Coinjoin pool	First transaction (txid)	Last transaction (txid)
Wasabi 1.x (zkSNACKs)	f250e997dc1a2d68861e03689d1709973e1964a62f929ba5727fe8607dafb676	390b971e0990b0689c306473dcb98b6b7481adf75d83652c30cfbc78912d9ed7
Wasabi 2.x (zkSNACKs)	d31c2b4d71eb143b23bb87919dda7fdfecee337ffa1468d1c431ece37698f918	0dc6180a7f6ab204cba382065c06c04f6fdd58846437af2d078b4f48eba25d95
Wasabi 2.x (post-zkSNACKs)	4315e96b05f9c87d96bcd9d35b520e3d2c4f4851558420774d2e2a88270d5cd4	d93758d59a9ec82d65fa7cb8d00fecb5ae655fbe9ad7e1208b8b3db0608f5814
Wasabi 2.x (kruw.io)	e8ab4e3f1c0133cc1cccf6f9f6dd499cb31fb07ff1b2e27e64b4b9d9750188ca	d1e7efee0071dd9e72481bf73ad35371b1f402fbf11a9a7354c129df5a773f44
Wasabi 2.x (gingerwallet)	75d060816ca08d067a91ba982e330aba7c5a2d50db2605403567989370120a66	8e7c637f8fb584ac618bb4fcbf8a08eb1534d9a288b9a6ad9f42e2a6afc0832f
Wasabi 2.x (opencoord.)	1c875d73ff53f221072432fdfd1a3f666a4a5e0dccb4021beb535537ccd36baa	d93758d59a9ec82d65fa7cb8d00fecb5ae655fbe9ad7e1208b8b3db0608f5814
Wasabi 2.x (wasabicoord.)	309f790cf0aec4a010698cb1adf4f9ce8f4d438b15cb9bdbfc97951fbef473d7	4432b1065163a1eafd151ffdf5f509e97a6402086c12c2c291a317481caf60b9
Wasabi 2.x (unknown_2024_e85631)	4315e96b05f9c87d96bcd9d35b520e3d2c4f4851558420774d2e2a88270d5cd4	427fee6b90d10f5b6204e7b1d3738081bda236549586a472be4338159d6ce8ff
Wasabi 2.x (unknown_2024_28ce7b)	28ce7b2815bce935b9f1ec7bb183f0d1365b5005789e6175d5771f72190a862f	1ac49264159f44468013ba101f32ae167cb3423d8fde022c86d7dff23ca2d9bb
Wasabi 2.x (coinjoin.nl)	1f2622282e05f3dd5a0af95b49fbee4f9c577abfc7edd4a362237ab9ab6c72bf	ab4cf228fc301d7d0c77e8d1d12d19ecf2d1f9b6699821f6bfffde45bea4f4d4
Whirlpool all (Samourai)	a554db794560458c102bab0af99773883df13bc66ad287c29610ad9bac138926	7369a8c414405f36ddddbb6d4c89b3680b214b50b2cf91eab0747a4aee6f8c6b
Whirlpool 5M	a554db794560458c102bab0af99773883df13bc66ad287c29610ad9bac138926	d30a07a865757ab4b1e1366cd8c07cc85daa10300ef77c025d2dcaedbf42a827
Whirlpool 1M	c6c27bef217583cca5f89de86e0cd7d8b546844f800da91d91a74039c3b40fba	8bc4ba7b1805c8a4b2f35e2f66b430192e6760cdc23dfef3403fcd9a1b7723c8
Whirlpool 50M	b42df707a3d876b24a22b0199e18dc39aba2eafa6dbeaaf9dd23d925bb379c59	c10267ce9e8d2fc09262d3743ed2bfa094c4e15fad9e2d479fd63ed7d21b8220
Whirlpool 100k	ac9566a240a5e037471b1a58ea50206062c13e1a75c0c2de3f21c7053573330a	7369a8c414405f36ddddbb6d4c89b3680b214b50b2cf91eab0747a4aee6f8c6b
Whirlpool all (Ashigaru)	737a867727db9a2c981ad622f2fa14b021ce8b1066a001e34fb793f8da833155	e6b289ac228a31a2c4a16bc976fdd828286f8190f50557e8e07998b8cf5759b9
Whirlpool 2.5M	737a867727db9a2c981ad622f2fa14b021ce8b1066a001e34fb793f8da833155	e6b289ac228a31a2c4a16bc976fdd828286f8190f50557e8e07998b8cf5759b9
Whirlpool 25M	7784df1182ab86ee33577b75109bb0f7c5622b9fb91df24b65ab2ab01b27dffa	b8f50518d2a799d1f2041e7e77fe3eed5d082d490023e3a2ebcd907aba87bd00

Table 4: Transaction IDs of first and last coinjoin attributed to given coordinator till 10th November 2025.

M Wasabi 2.x statistics

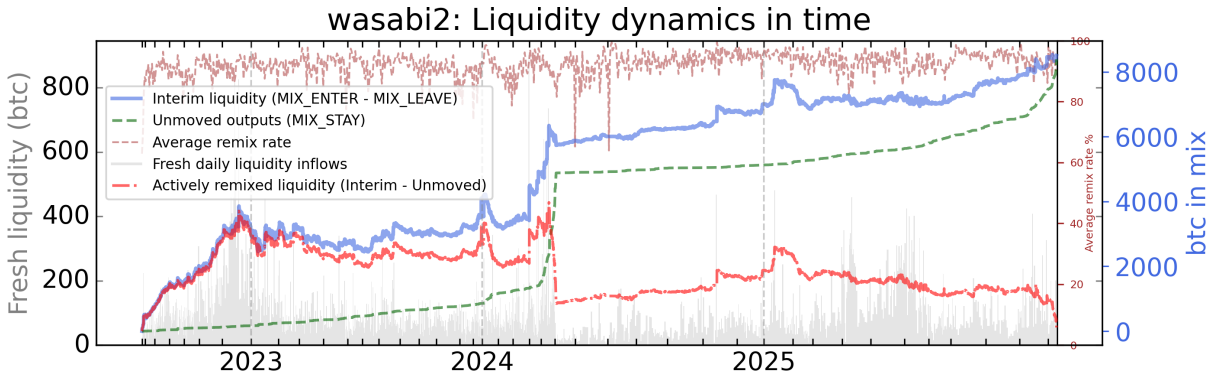

Figure 23: Cumulative liquidity dynamics for all Wasabi 2.x coordinators.

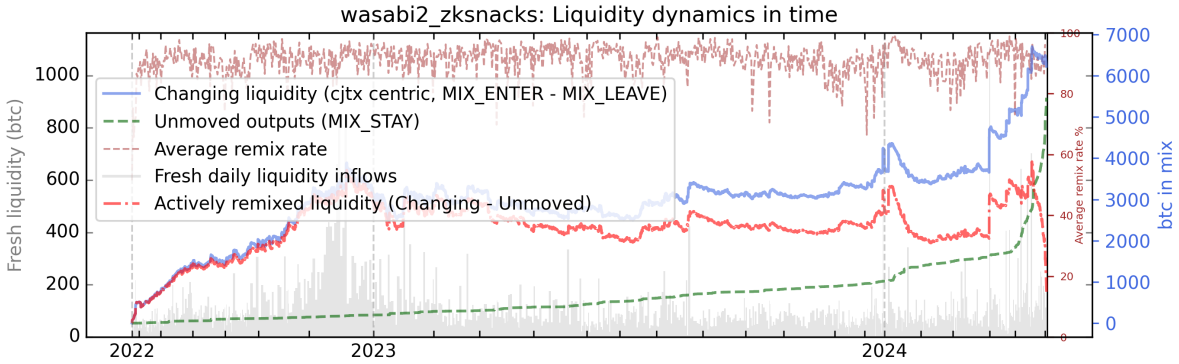

Figure 24: Cumulative liquidity dynamics for Wasabi 2.x zkSNACKs coordinator.

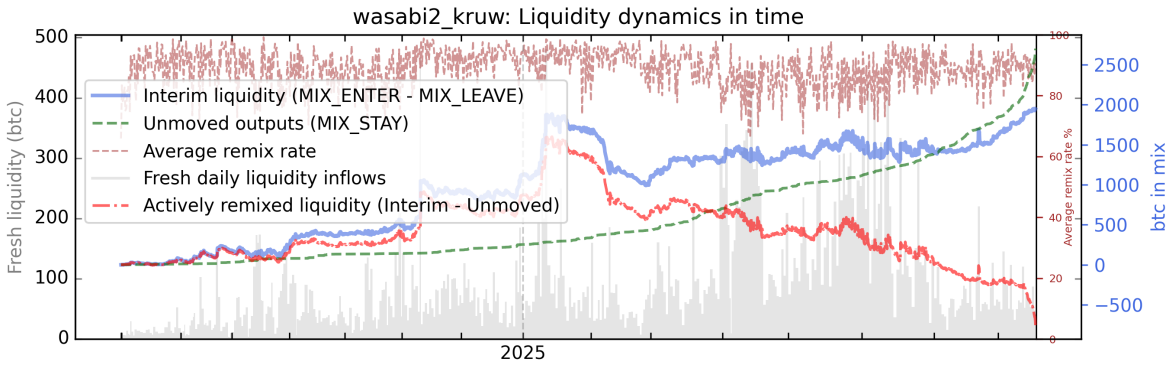

Figure 25: Cumulative liquidity dynamics for Wasabi 2.x kruw.io coordinator. Note that actively remixed liquidity (red dashed line) may take negative values when incoming liquidity is smaller than outgoing liquidity. This occurs because input liquidity previously already mixed under *zkSNACKs* or another WW2 coordinator is intentionally not counted as fresh—otherwise, overall WW2 design liquidity would be artificially inflated by wallets changing coordinators.

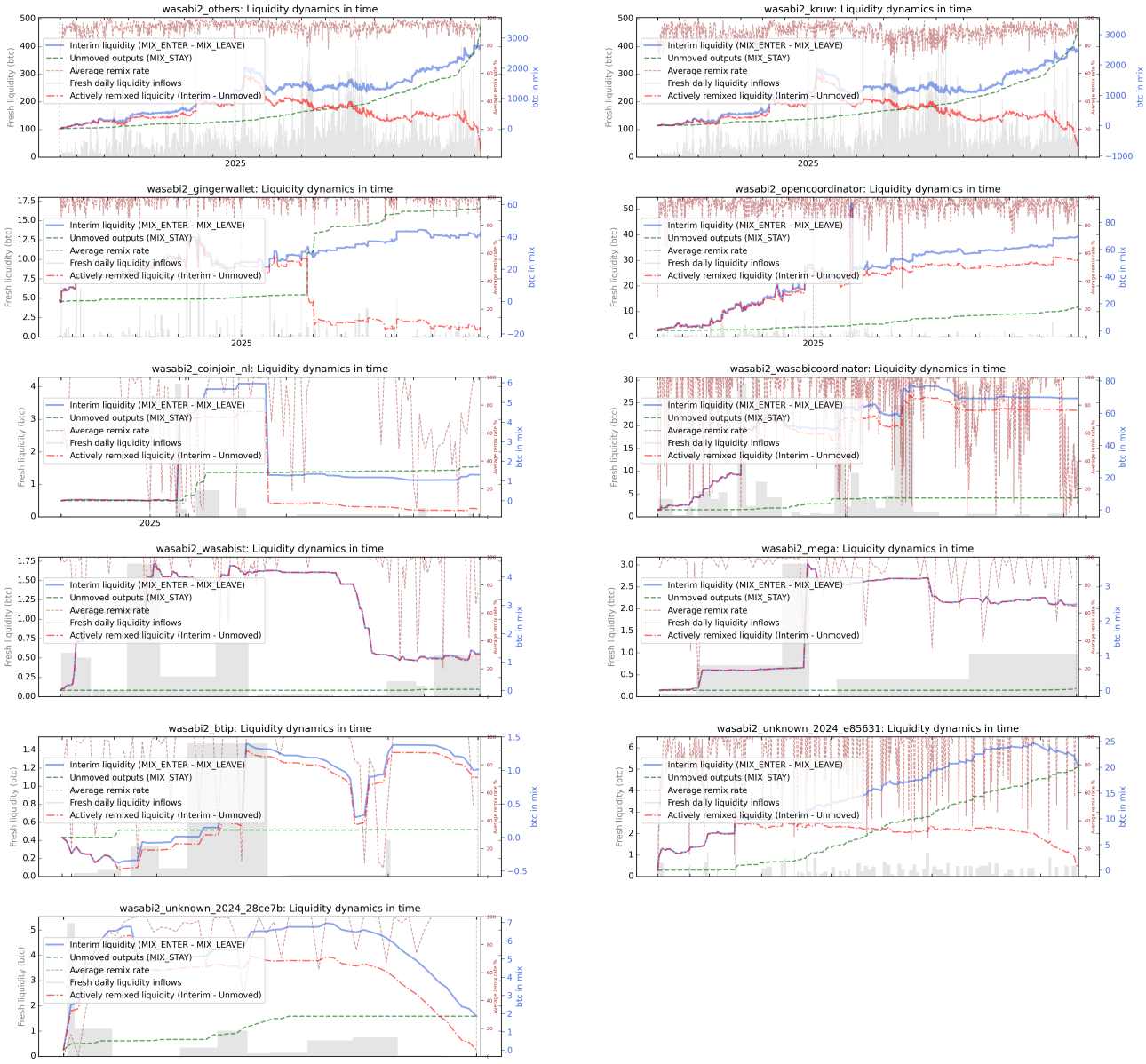

Figure 26: Liquidity dynamics for all non-trivial detected WW2 coordinators. Note that actively remixed liquidity (red dashed line) may take negative values when incoming liquidity is smaller than outgoing liquidity. This occurs because input liquidity previously already mixed under *zkSNACKs* or another WW2 coordinator is intentionally not counted as fresh—otherwise, overall WW2 design liquidity would be artificially inflated by wallets changing coordinators. Both *kruw.io* and *gingerwallet* illustrate this case.

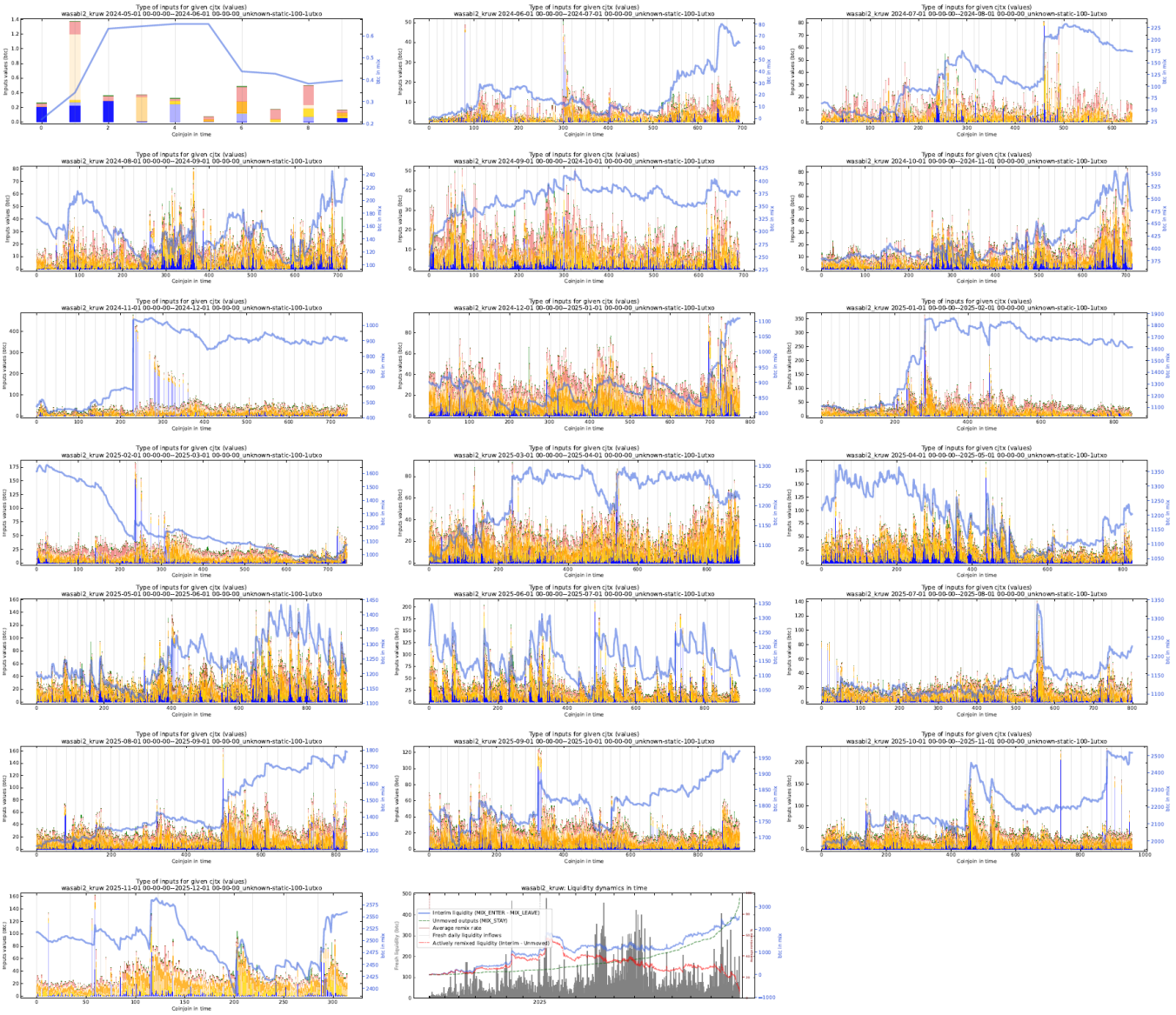

Figure 27: WW2 *kruw.io* monthly liquidity changes with focus on value of inputs without normalization. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

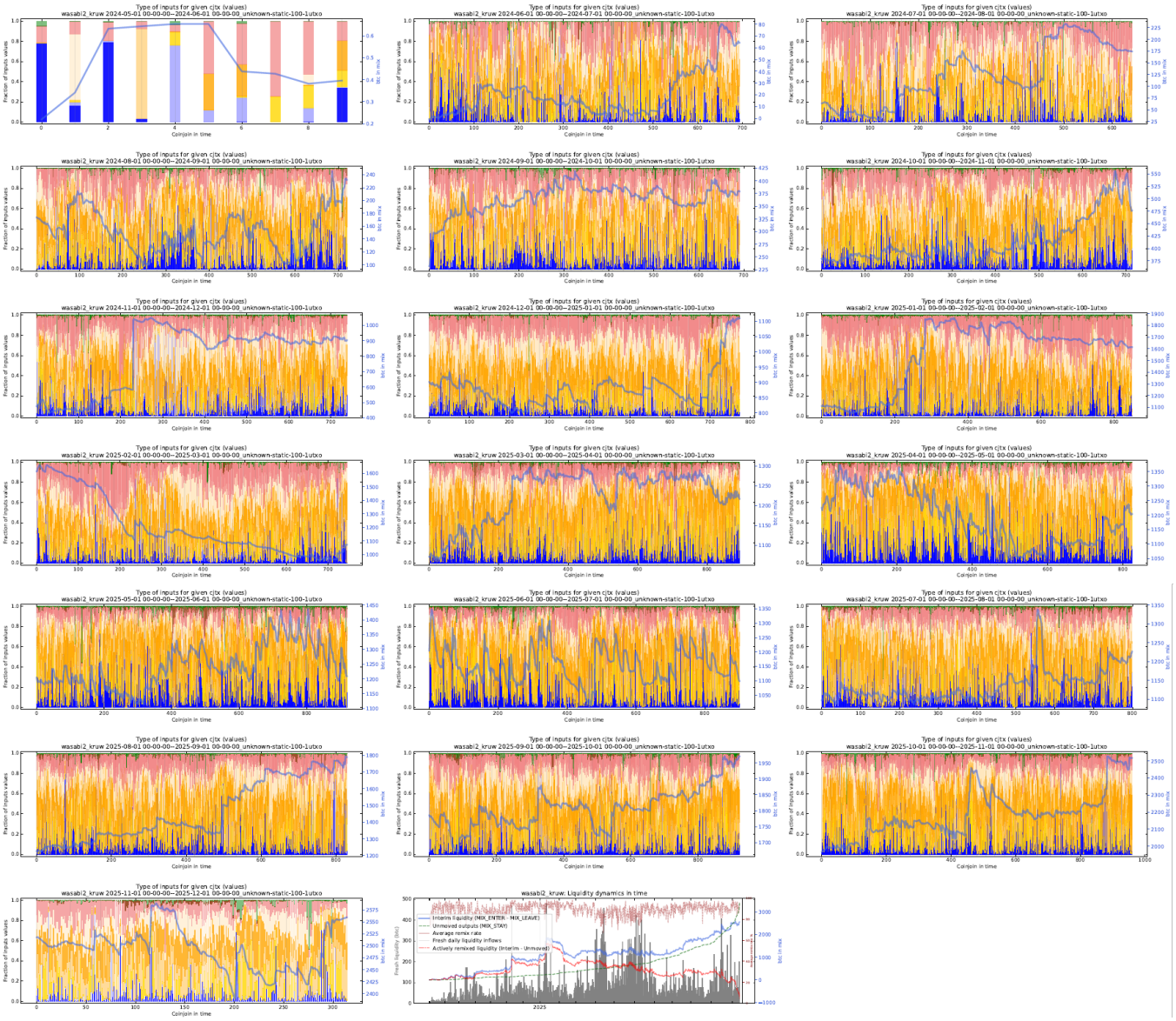

Figure 28: WW2 *kruw.io* monthly liquidity changes with focus on number of inputs with normalization. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

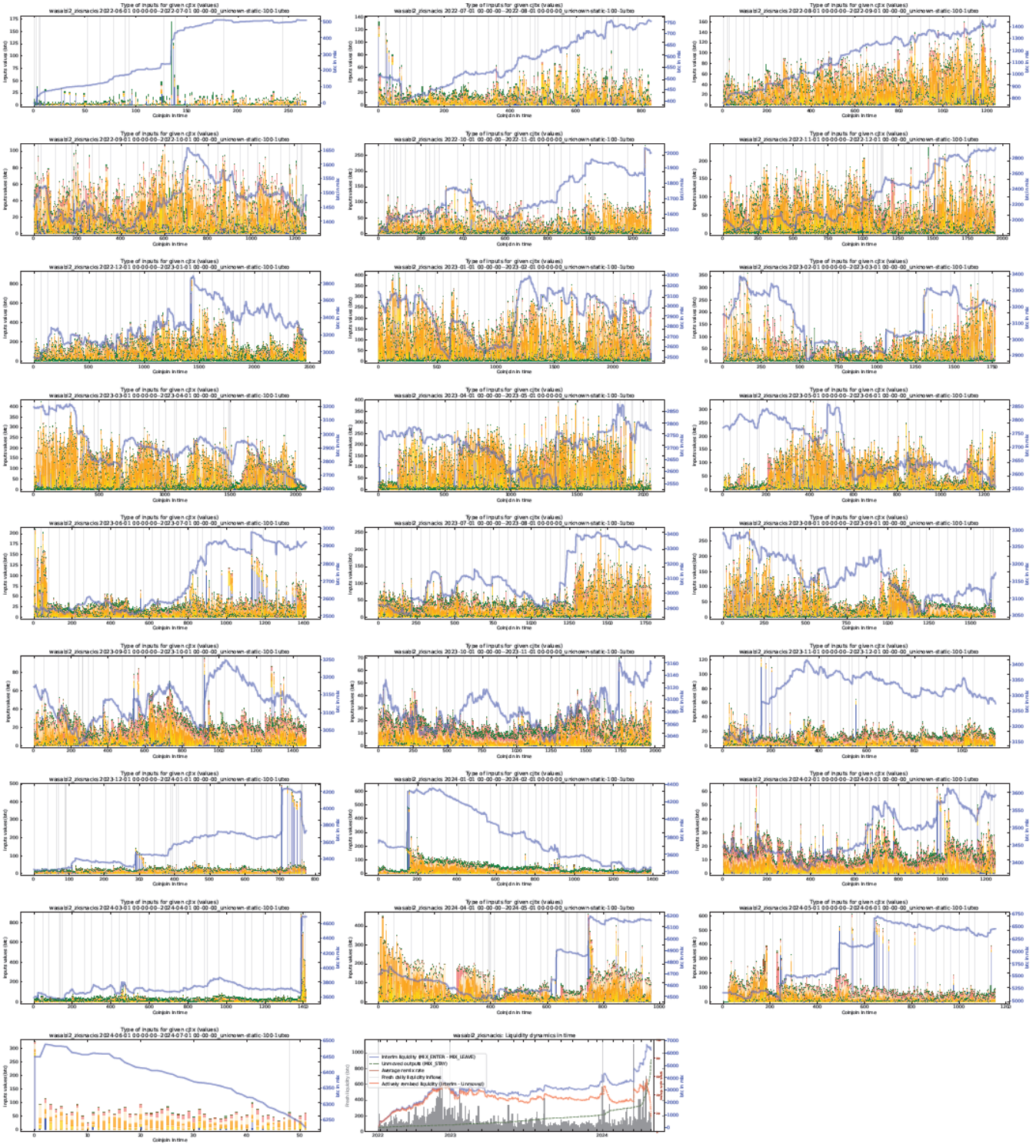

Figure 29: WW2 *zkSNACKs* monthly liquidity changes with focus on value of inputs without normalization. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

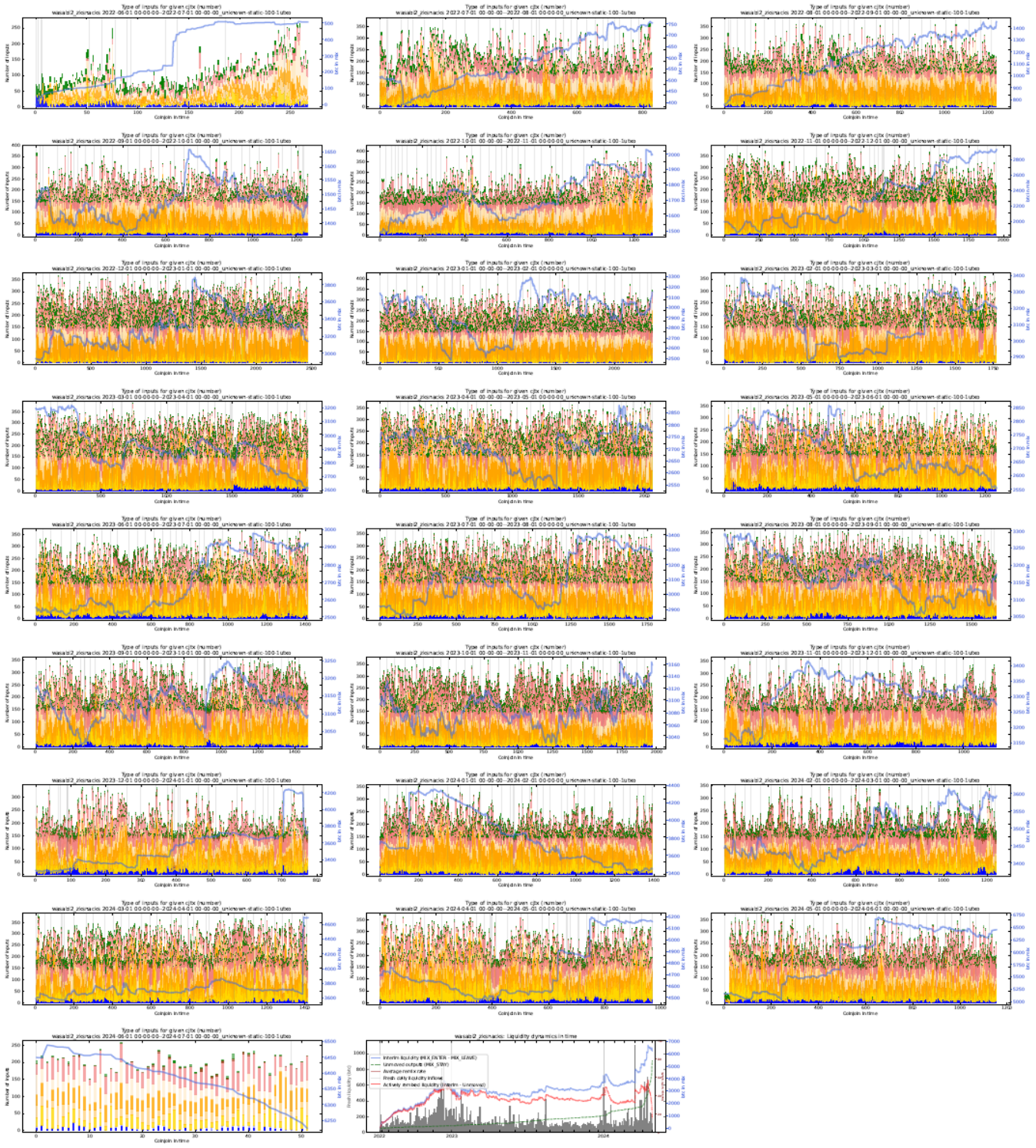

Figure 30: WW2 *zkSNACKs* monthly liquidity changes with focus on the number of inputs without normalization. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

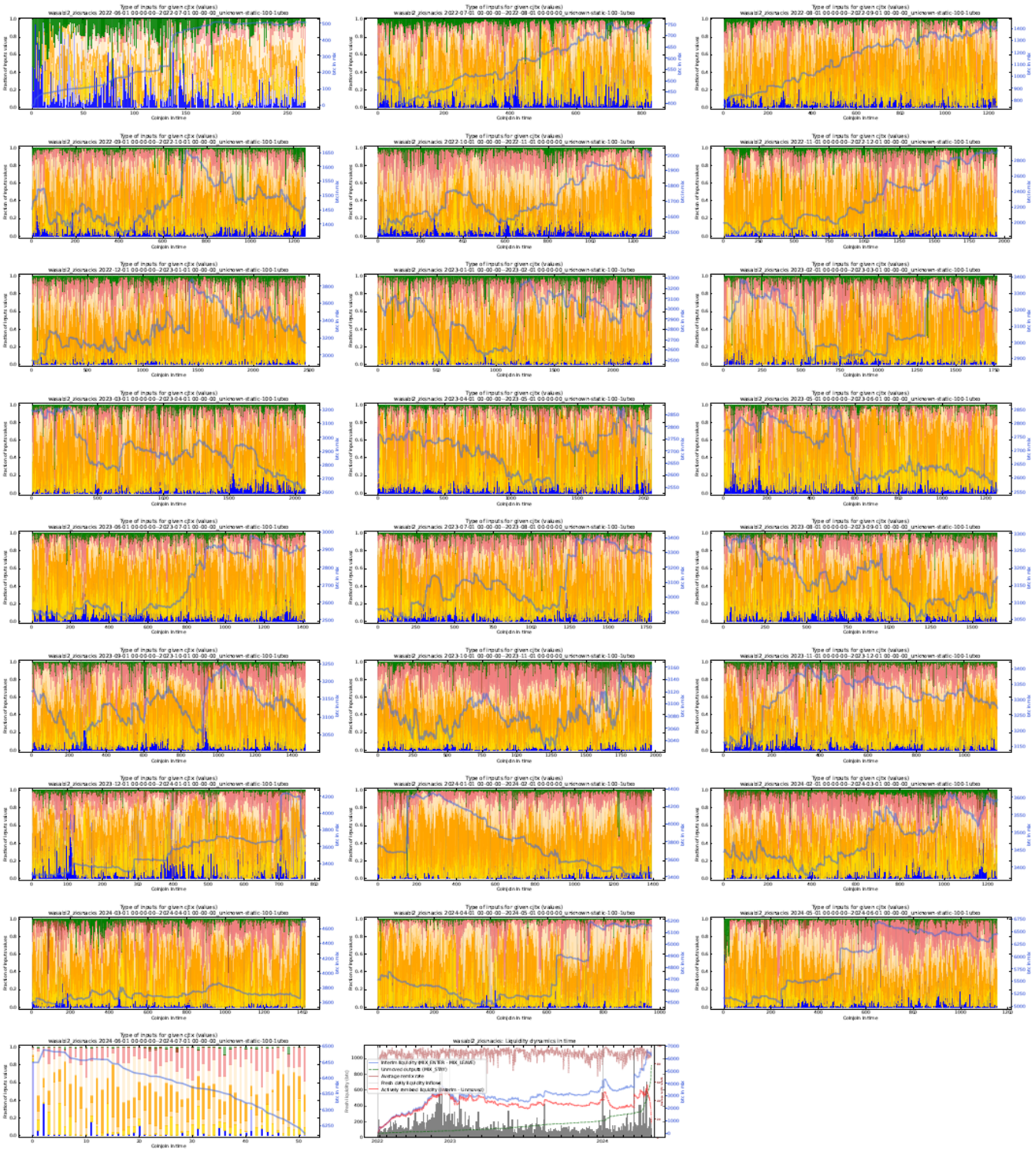

Figure 31: WW2 *zkSNACKs* monthly liquidity changes with focus on value of inputs with normalization, highlighting new fresh inflows. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

N Wasabi 1.x statistics

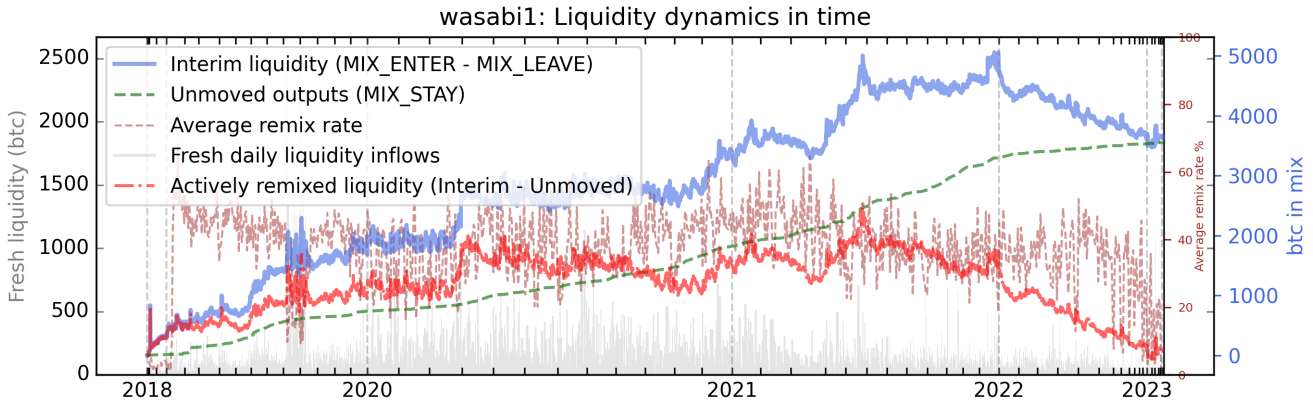

Figure 32: Cumulative liquidity dynamics for Wasabi 1.x

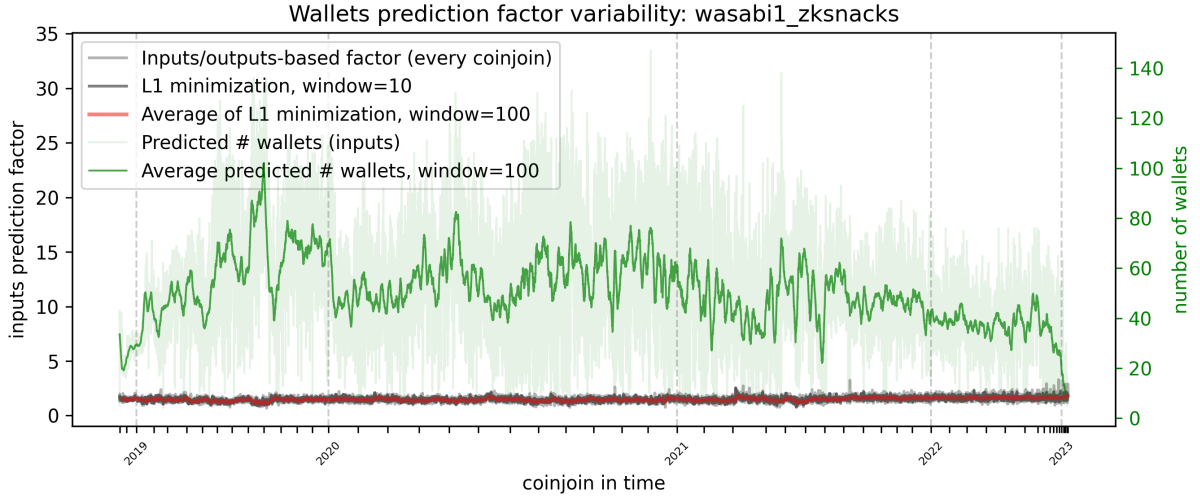

Figure 33: Variability of inputs-based wallets prediction factor of number of active wallets for WW1 *zkSNACKs* coordinator.

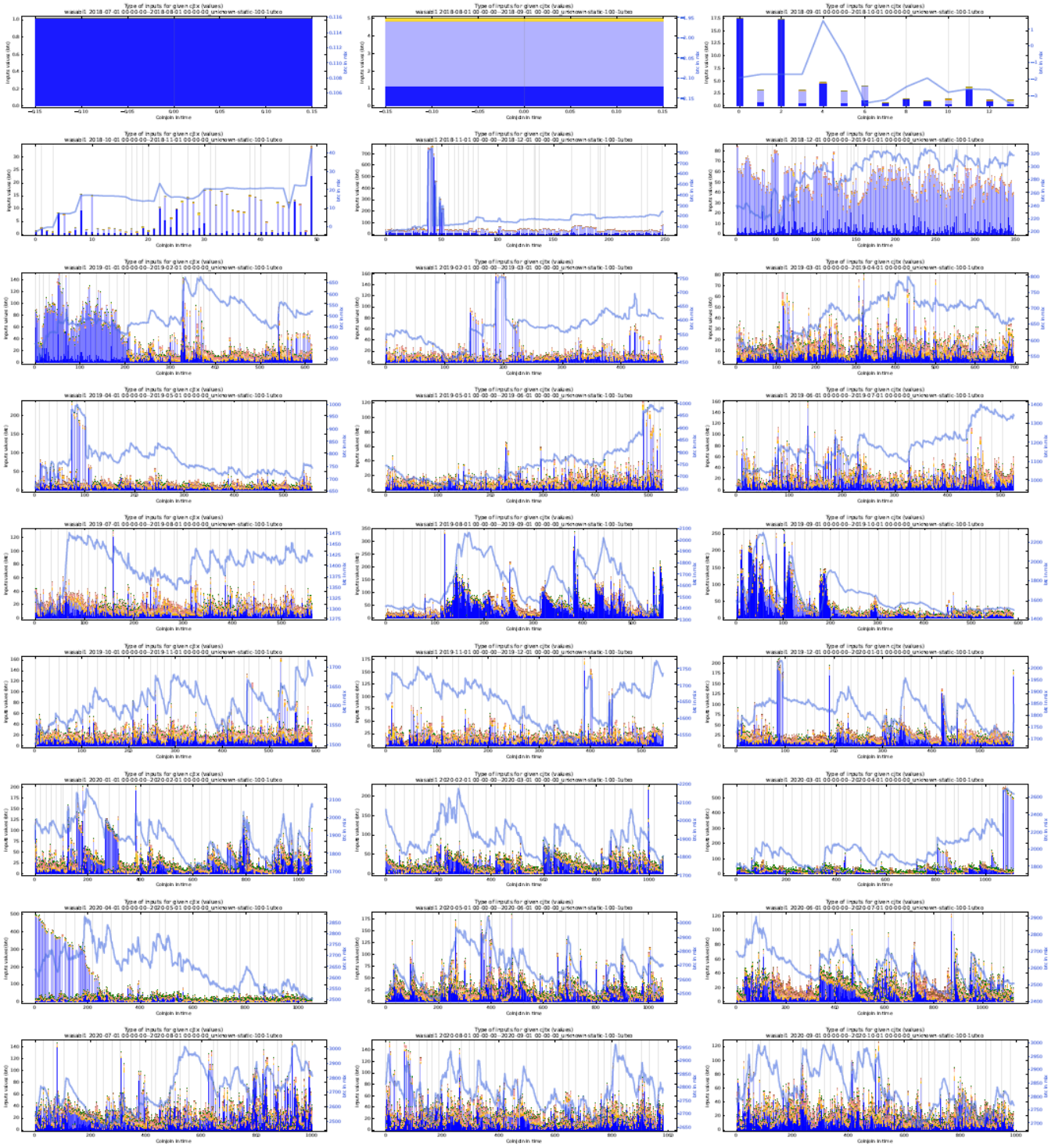

Figure 34: WW1 *zkSNACKs* monthly liquidity changes with focus on value of inputs without normalization. Period 2018-07 – 2020-09, continues on Figure 35. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

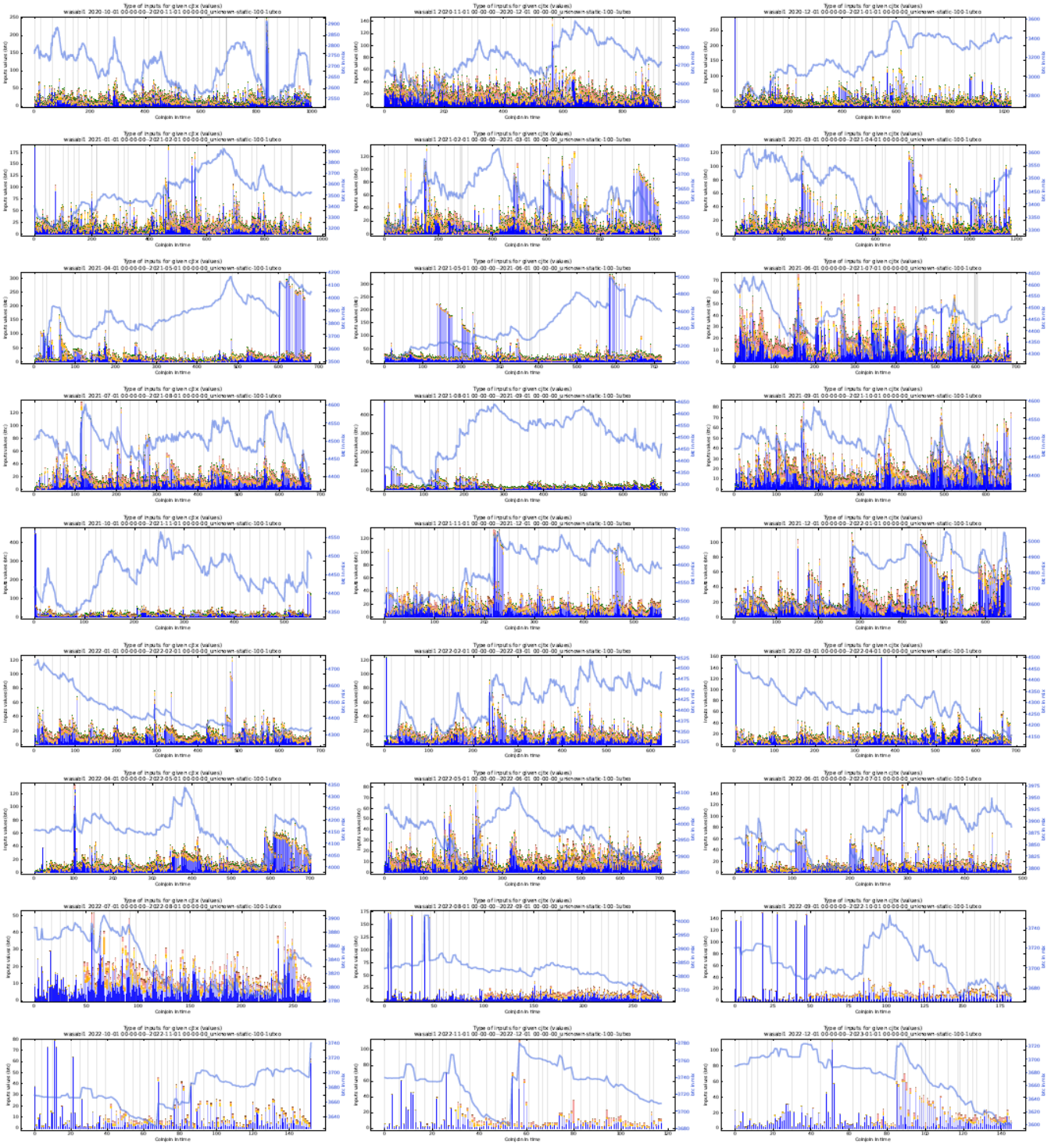

Figure 35: WW1 *zkSNACKs* monthly liquidity changes with focus on value of inputs without normalization. Period 2020-10 – 2022-12, continues on Figure 36. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

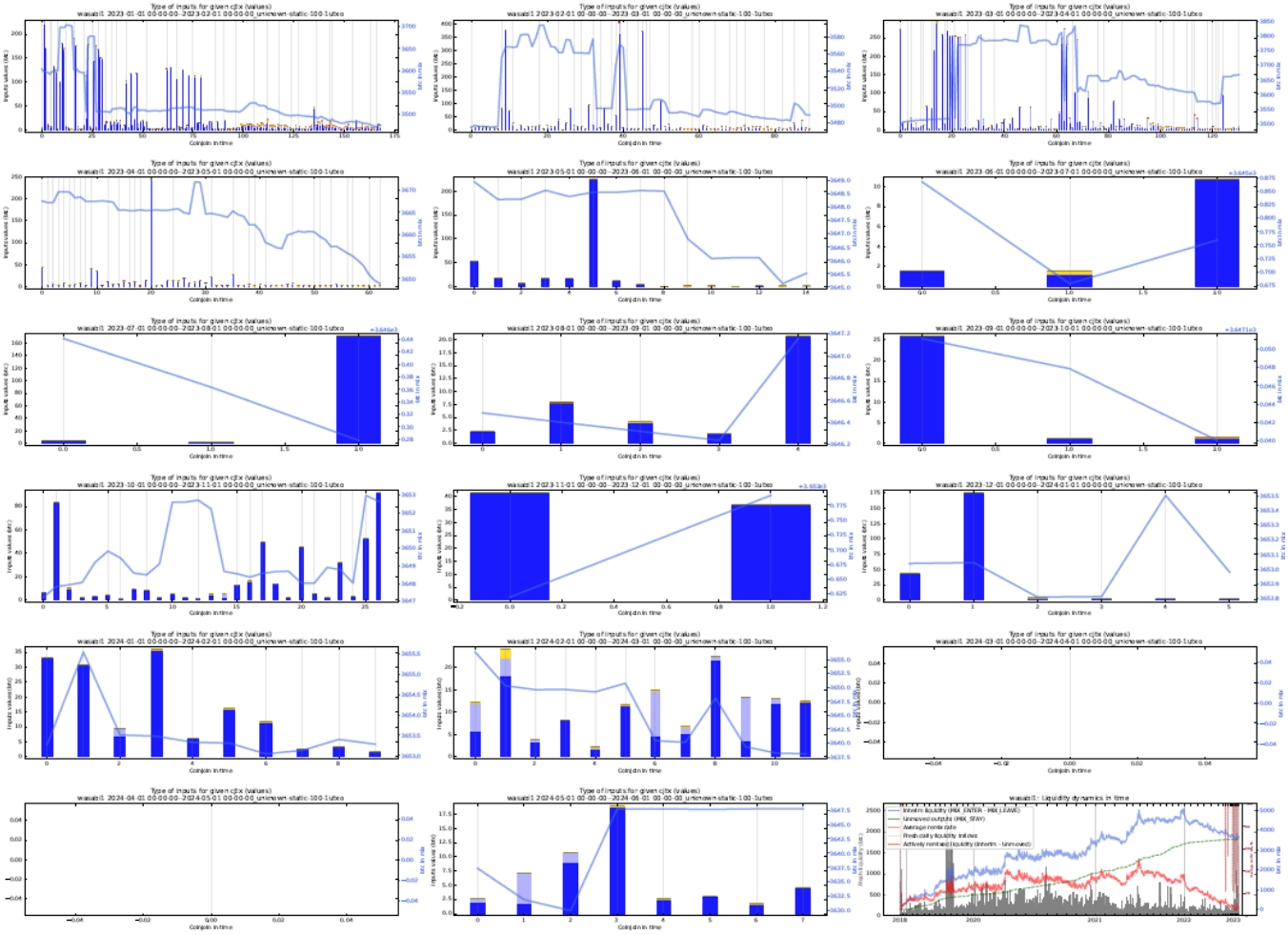

Figure 36: WW1 *zkSNACKs* monthly liquidity changes with focus on value of inputs without normalization. Period 2023-01 – 2023-05, till the coordinator was voluntarily shut down in favor of WW2 design with *zkSNACKs* coordinator. High-resolution version available at https://crocs-muni.github.io/coinjoin/.

O Whirlpool statistics

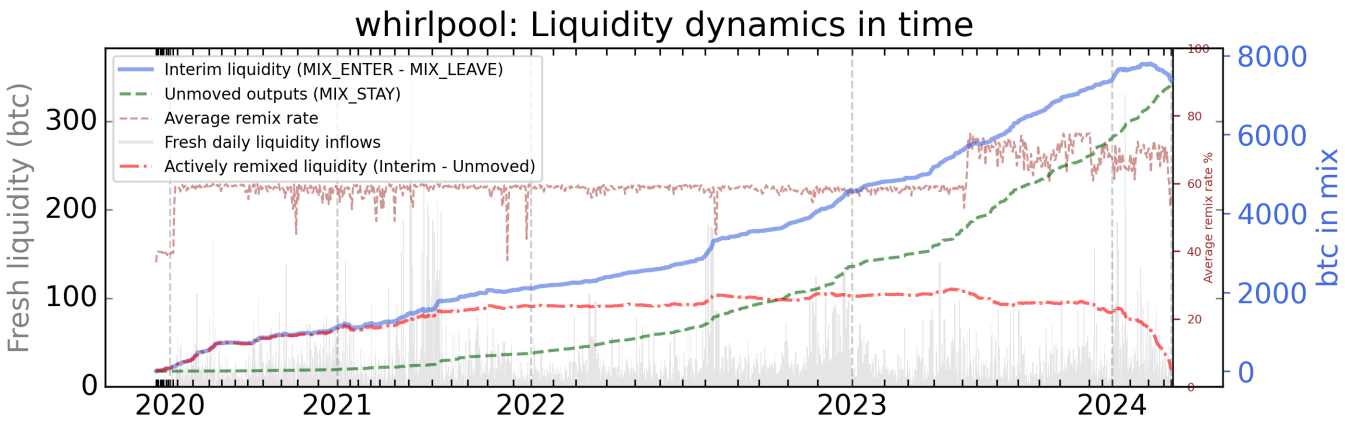

Figure 37: Cumulative liquidity dynamics for Samourai Whirlpool and Ashigaru Whirlpool with all pools combined.

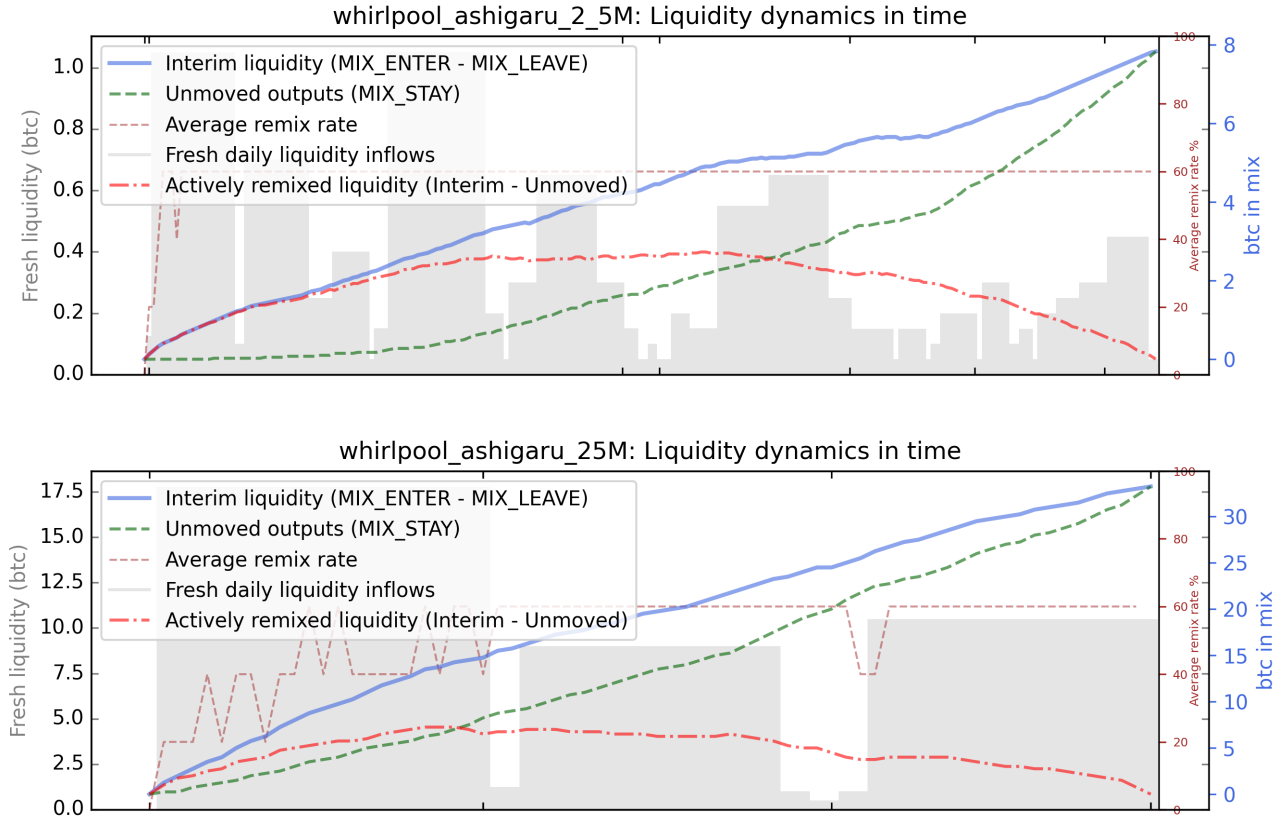

Figure 38: Liquidity dynamics for Ashigaru Whirlpool pools started from 31st May 2025 with sizes of 2.5M and 25M satoshis.

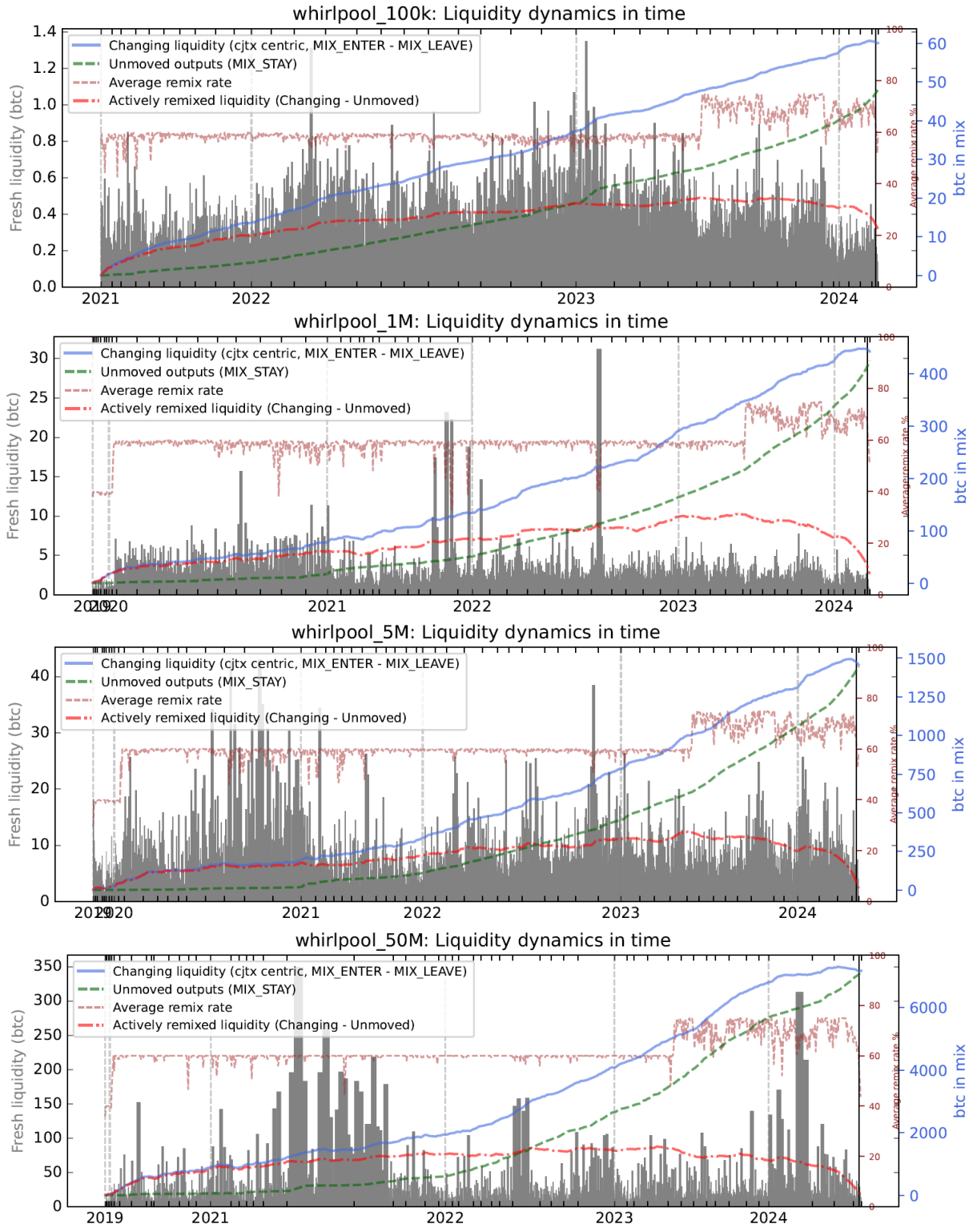

Figure 39: Liquidity dynamics for Samourai Whirlpool pools with sizes of 100k, 1M, 5M and 50M satoshis separately. The final lower value for unmoved outputs than cjtx centric (blue) visible (especially for pools with smaller sizes) with respect to changing liquidity is caused by the mining fees spent.