Unnecessary Input Heuristics & PayJoin Transactions

Simin Ghesmati[1,3], Andreas Kern[2,3], Aljosha Judmayer[2,3], Nicholas Stifter[2,3], and Edgar Weippl[2,3]

[1] Vienna University of Technology, Austria
[2] University of Vienna, Austria
[3] SBA Research, Vienna, Austria
`Email:(firstletterfirstname)(lastname)@sba-research.org`

This work published on HCI International 2021

Abstract. Over the years, several privacy attacks targeted at UTXO-based cryptocurrencies such as Bitcoin have been proposed. This has led to an arms race between increasingly sophisticated analysis approaches and a continuous stream of proposals that seek to counter such attacks against users' privacy. Recently, PayJoin was presented as a new technique for mitigating one of the most prominent heuristics, namely *common input ownership*. This heuristic assumes that the inputs of a transaction, and thus the associated addresses, belong to the same entity. However, a problem with PayJoin is that implementations can accidentally reveal such transactions if the corresponding inputs from involved parties are not chosen carefully. Specifically, if a transaction is formed in a way such that it contains seemingly unnecessary inputs, it can be identified through so-called *unnecessary input heuristic (UIH)*. What is not yet clear is the impact of naive coin selection algorithms within PayJoin implementations that may flag such transactions as PayJoin. This paper investigates the resemblance of PayJoin transactions to ordinary payment transactions by examining the significance of the unnecessary input heuristic in transactions with more than one input and exactly two outputs which is the common template of recent PayJoin transactions.

Keywords: PayJoin · Bitcoin · privacy · blockchain · mixing · unnecessary input heuristic · optimal change heuristic.

1 Introduction

Blockchain-based cryptocurrencies such as Bitcoin have gained significant interest by a wider audience in recent years. Hereby, the topic of transaction privacy has received considerable attention as research clearly highlights that storing every transaction in the network within a publicly accessible ledger can have a serious effect on user privacy. Previous studies [1,2] report some of the possible attacks that can reveal the identities of different entities and effectively find their relationships within UTXO(unspent transaction output)-based blockchains such as Bitcoin. Several techniques have been proposed to remedy these privacy issues. In Bitcoin, the amount (also referred to as coins) associated with an address can be transferred to another address through a transaction. Transactions consist of input and output addresses. Each input address should be signed by its private key to unlock transferring the coins from that address. It is assumed that all the inputs in a transaction are controlled by the same user. This leads to a so-called *common input ownership heuristic* [1] which helps to cluster all the addresses that belong to the same user. To prevent effectiveness of this heuristic, CoinJoin was proposed by G. Maxwell in 2013 in which the users jointly create a transaction with their inputs and their desired outputs and then each of the users separately signs her input. Users have to send the same amount of coins to the desired outputs to prevent any linkage between the inputs and the outputs, which in turn leads to the distinguishability of these equal-sized output transactions in the blockchain. In recent years, the PayJoin [3,4,5] protocol, which follows the basic idea of CoinJoin, has been proposed to enhance the privacy of Bitcoin transactions, whereby the intended recipient of a transaction adds some of her *own* unspent transaction outputs to the inputs of the sender's transaction to break the so-called *common input ownership heuristic* [1]. The technique has been proposed as a Bitcoin Improvement Protocol BIP78 [6]; however, a naive implementation of the protocol, specifically in regard to the coin selection by participants, has the potential to flag such transactions as a PayJoin.

This paper evaluates the significance of *unnecessary input heuristic* on PayJoin transactions and discusses possible solutions to better blend in these transactions. In particular, the contributions of this paper are as follows:
- We compare the different definitions of unnecessary input heuristic (UIH).
- We provide an empirical analysis of the different UIH approaches.
- We extend upon existing discussions and describe possible countermeasures for PayJoin technique.

The remaining part of the paper is structured as follows: We first provide definitions and background information on PayJoin transactions in Section 2. Section 3 examines the definition of unnecessary input heuristic, reports extracted transaction statistics from Bitcoin, analyzes them via this heuristic. We then discuss research challenges specific to the PayJoin protocol in Section 4.

2 PayJoin transactions

The concept of PayJoin was first proposed in a blog-post by BlockStream, called Improving Privacy Using Pay-to-EndPoint (P2EP) [3]; shortly after, BustaPay [4] was proposed in the Linux Foundation, Gibson provided more details under the name of PayJoin [5]. PayJoin solves the distinguishability of equal-size CoinJoin transactions by adding at least one UTXO of the recipient to the UTXO inputs of the transaction (Figure 1), which provides plausible deniability. Simplified, it can be considered as performing a CoinJoin while paying someone else. The protocol effectively breaks one of the most prominent heuristics that can be employed to de-anonymize Bitcoin users, namely common input ownership. By breaking this heuristic, the utilization of the PayJoin protocol by some users can also provide privacy to other users. Further, the protocol also intends to hide the true payment amount, as the total output of the transaction will be the sum of the payment amount and the recipient's input amounts.

To run the protocol [6], the recipient (Bob) sends his address and the amount to the sender (Alice), using BIP21 URI. The sender creates and signs a transaction (original transaction) in which she sends the specified amount to the recipient's address. She also provides her change address to receive the remainder, and then sends the transaction to the recipient. The recipient checks the transaction and creates a new transaction (PayJoin proposal) by appending his inputs to the transaction created by the sender. The recipient then alters the output amount by adding his input amounts. He signs his inputs and sends this PayJoin proposal to the sender. The sender checks and signs the PayJoin transaction and broadcasts it to the network. The recipient is also able to broadcast the original transaction if there was any problem in creating the PayJoin transaction. At the time of writing, the protocol has been implemented by Joinmarket wallet, Samourai wallet (Stowaway), Wasabi wallet, Blue wallet and BTCPay. Users can create their own stores in BTCPay to receive PayJoin transactions (e.g., for selling their services). The senders (or buyers) can pay to the recipients through the aforementioned wallets that support PayJoin transactions. Currently, PayJoin transactions are formed as multiple inputs and exactly two outputs, which we consider as interesting transactions in this paper.

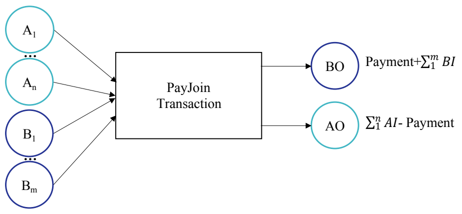

Fig. 1. A sample PayJoin transaction

3 Optimal Change/Unnecessary Input Heuristic

Whenever the sum of the inputs is larger than the payment amount, a so-called change address is created in Bitcoin to return the remainder of the coins to the sender [1]. Over the years, several heuristics have been implemented to aid address attribution. In this section, we want to look at one of the heuristics, which can be directly used to identify PayJoin transactions, amongst all transactions with more than one input and exactly two outputs.

UIH1 (Optimal change address): The *optimal change* heuristic was introduced to detect the change address of a transaction. It has been implemented by blockchain analysis tools such as blocksci [2] to cluster addresses. This heuristic flags the smallest output as a change address if it is smaller than the smallest input [2].

UIH2 (Unnecessary Input): This heuristic flags transactions as abnormal, if the largest output could be paid without the smallest input.

In practice, there exist several nuanced variants of UIH1 and UIH2, which we discuss in the following.

BlockSci UIH1 [2]: *"If there exists an output that is smaller than any of the inputs it is likely the change."*[4]

BlockStream UIH1[5]: *"This heuristic gives an indication that one output is more likely to be a change because some inputs would have been unnecessary if it was the payment."*

Gibson UIH1 [5]: *"One output is smaller than any input. This heuristically implies that output is not a payment, and must therefore be a change output."*

BlockSci UIH2 [2]: *"If a change output was larger than the smallest input, then the coin selection algorithm would not need to add the input in the first place."*[6]

BlockStream UIH2: *"If the sum of the inputs minus minimum input covers the larger output and transaction fee, the transaction has seemingly unnecessary inputs that are not typically added by consumer wallet software with a less sophisticated coin selection algorithm."*

Gibson UIH2 [5]: *"One input is larger than any output. This heuristically implies that this is not a normal wallet-created payment."*

Gibson, A. [5] and BlockStream definitions need double-checking for UIH1 and UIH2; however, UIH2 by [5] does not cover the situation when the subset sum of the inputs is enough for paying the larger output. There is also a degree of uncertainty over the definition of UIH2, as different coin selection algorithms or input consolidation can violate this heuristic [7], however, creating those kinds of transactions by PayJoin protocol has the potential to make these kinds of transactions suspicious. Avoiding UIH1 and UIH2 is an open question amongst PayJoin developers, and what is not yet clear is the anonymity set that can be achieved by avoiding these heuristics. The discussion among the developers refers to the statistics that were provided on gist and then published in [5]. The statistics provide the number of transactions with more than one input and exactly two outputs for 885 blocks in one week of the December 2018. To clarify how these transactions appear in practice, we refer to these real PayJoin transactions that can be categorized by UIH1[7] and UIH2[8]. As can be seen, one performed such as normal payment transactions (UIH1) and the other has unnecessary input (UIH2)[9]. PayJoin technique is recently implemented and there is not enough ground truth, however, in our manual inspection to find BTCPay PayJoin transactions and in the transactions that we created using BTCPay, we

[4] We consider the BlockSci definition in a simpler form, ignoring the transaction fee for data categorization.

[5] `https://github.com/Blockstream/esplora/blob/cbed66ecee9f468802cf1f073c204718beac30d7/client/src/lib/privacy-analysis.js#L47-L70`

[6] A comprehensive definition can be considered as $(sum(in) - min(in) >= sum(out) - min(out) + TX.fee)$

[7] `https://blockstream.info/tx/8cb0af96f1a2693683621758acbf3b7a7ad69a69672c61e144941d666f72da2a`

[8] `https://blockstream.info/tx/58d68b22ab96b87a11c1fbd3090fee23f96f71a4115f96210ba776d0ae7d8d55`

[9] In the second transaction, different nSequence fields also reveal that the inputs were added by different wallets, however, wallet fingerprinting is beyond the scope of this paper.

Algorithm 1 BlockSci UIH1 and UIH2 according to BlockSci !UIH1

```
if $min(out) < min(in)$ then
    Transaction is UIH1
else
    Transaction is UIH2
end if
```

Algorithm 2 BlockStream UIH1 and UIH2

```
if $(Sum(in) - min(in) >= max(out) + TX.fee)$ then
    Transaction is UIH2
else if $(Sum(in) - min(in) >= min(out) + TX.fee)$ then
    Transaction is UIH1
end if
```

Algorithm 3 Gibson UIH1 and UIH2

```
if $min(out) < min(in)$ then
    Transaction is UIH1
end if
if $max(in) > max(out)$ then
    Transaction is UIH2
end if
```

found that BTCPay does not prevent UIH2 and some of the transactions are formed such that they can be flagged as UIH2.

3.1 Transactions statistics via the Unnecessary Input Heuristic

We Parsed Bitcoin blockchain and extracted all the interesting transactions over a week in September from 2009 to 2020, and then apply different definitions to shed light on the significance of avoiding UIH1 and UIH2 in PayJoin transactions as well as to compare the statistics obtained by different definitions. We Parsed Bitcoin blockchain and extracted all the transactions with more than one input and exactly two outputs for 885 blocks in one week of the September from 2009 to 2020, the statistics of which are indicated in figure 2. Due to practical constraints, this paper cannot provide comprehensive data for all the transactions which are performed in the Bitcoin blockchain.

As illustrated in the table, the classical payment transactions with exactly two outputs decreased in recent years. The other point that can be achieved is that interesting transactions are almost 20% of all transactions on average. PayJoin at this stage can only be hidden in this set of transactions, which can be considered as the possible anonymity set. Further, this anonymity set is almost 15% in September 2020 which indicates a smaller anonymity set in the year that PayJoin was implemented by most of the wallets. We then analyzed the interesting transactions through SC−UIH1 (BlockSci), G−UIH1 (Gibson [5]), ST−UIH1 (BlockStream), SC−UIH2 (BlockSci !UIH1), ST−UIH2 (BlockStream), and G−UIH2 (Gibson [5]). We also extracted the data for 885 blocks in December 2018 to compare our results with the statistics in [5].

We applied algorithms 1, 2, and 3 to categorize transactions by different definitions of UIH1 and UIH2. The results of our research are reported in table 1 which shows a breakdown of transactions by the different UIH1 and UIH2 categories. As can be seen in the selected blocks during September 2020, 64.8% of

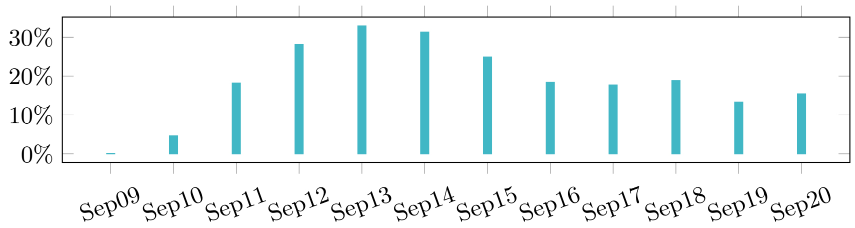

Fig. 2. Percentage of interesting transactions from total transactions

Table 1. UIH1 and UIH2 transactions

885 Blocks Start from	SC−UIH1[‡] G−UIH1[◇]	ST−UIH1[×]	SC−UIH2	ST−UIH2	G−UIH2	SC−UIH1% G−UIH1%	ST−UIH1%	SC−UIH2%	ST−UIH2%	G−UIH2%
3-Dec-18	145264	145003	122828	122684	83513	54.20%	54.09%	45.80%	45.76%	31.20%
9-Sep-09	1	1	0	0	0	100.00%	100.00%	0.00%	0.00%	0.00%
9-Sep-10	56	56	33	33	16	62.90%	62.92%	37.10%	37.08%	18.00%
9-Sep-11	5258	5282	2323	2288	466	69.40%	69.67%	30.60%	30.18%	6.10%
9-Sep-12	21107	20982	32164	32137	21551	39.60%	39.39%	60.40%	60.33%	40.50%
9-Sep-13	44308	44277	27448	27258	15539	61.70%	61.70%	38.30%	37.99%	21.70%
9-Sep-14	64926	65432	55308	54653	36083	54.00%	54.42%	46.00%	45.46%	30.00%
9-Sep-15	100210	100699	106913	105518	84373	48.40%	48.62%	51.60%	50.94%	40.70%
9-Sep-16	133802	133345	82436	82359	57103	61.90%	61.67%	38.10%	38.09%	26.40%
9-Sep-17	124684	123971	90398	90197	60583	58.00%	57.64%	42.00%	41.94%	28.20%
9-Sep-18	140864	140474	112492	112428	74070	55.60%	55.45%	44.40%	44.38%	29.20%
9-Sep-19	144879	144143	88588	88544	69510	62.10%	61.74%	37.90%	37.93%	29.80%
9-Sep-20	171532	170790	93180	93120	75295	64.80%	64.52%	35.20%	35.18%	28.40%
Average						58.04%	57.98%	41.96%	41.77%	27.18%

[‡] SC stands for BlockSci. [×] ST stands for BlockStream. [◇] G stands for Gibson.

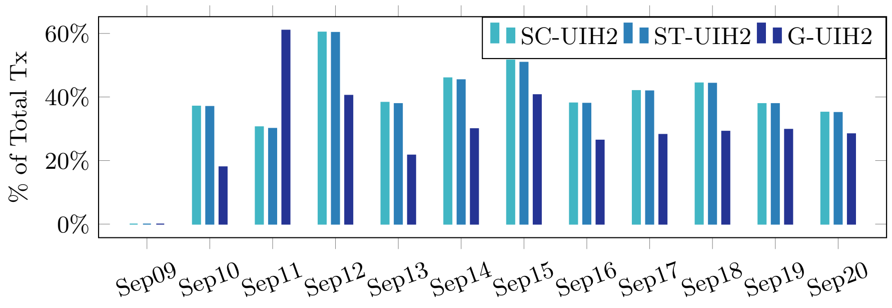

Fig. 3. Percentage of UIH2 transactions

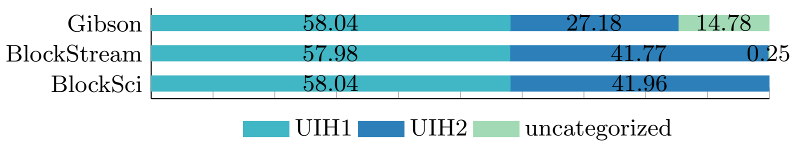

Fig. 4. % of transaction categories by BlockSci, BlockStream, and Gibson definitions

transactions are categorized as SC−UIH1 transactions, in which the change address is identifiable while 35.2% are SC−UIH2 which contain unnecessary input. Due to the small number of transactions in the selected data during September 2009, we exclude this year in the following statistical reports. The results obtained from the preliminary analysis show that on average nearly 58.04% of transactions are categorized as SC−UIH1 compared to around 41.96% as

SC−UIH2. An interesting aspect we observe is the high ratio of SC−UIH2 and ST−UIH2 transactions, which shows almost 40% of transactions did not follow normal coin selection algorithms and were created with unnecessary input. Figure 3 illustrates the chart of UIH2 transactions by different algorithms. The peak in September 2012 for the transactions that are categorized as SC−UIH2 and ST−UIH2 (60.40% and 60.33%, respectively) could be an interesting point to investigate and seek the reason for creating lots of transactions with unnecessary input during that time. From Figure 4 we can see that 57.98% and 41.77% of the transactions are categorized by ST−UIH1 and ST−UIH2, while 0.25% fit in none of the categories, which shows that Blockstream algorithm can not categorize all the transactions. The high ratio of uncategorized transactions (14.78%) by Gibsons' definition is as a result of considering only the largest input, instead of a subset sum of the inputs in UIH2 definition, which leads to flagging only 27.18% of the transactions as UIH2.

4 Discussion

At the time of writing, implementations of the PayJoin protocol remain relatively new and we conjecture that they are not yet used much in practice; thus, the data-set will likely not include many PayJoin transactions that are created as UIH2 and therefore in theory identifiable. Due to the large ratio of UIH2 transactions, the small number of PayJoins have not led to a bias in the last evaluation period (Sep 20), where they could have occurred. This might change if the reason for the high ratio of UIH2 transactions can be identified. A manual investigation of some of the UIH2 transactions revealed that there are transactions which can be categorized as internal address reuse, i.e., spending the UTXOs of the same address. This circumstance can separate these transactions from others, indicating that they are not PayJoin transactions. Further analysis is required to investigate the high number of UIH2 transactions. One of the possible countermeasures to avoid UIH2 in PayJoin transactions would be adding the input by the recipient such that it overtakes the change address of the sender (AO in Figure 1); in this manner, the minimum output would be less than the minimum input which results in creating the transaction as an ordinary one.

On the one hand, the PayJoin protocol has the potential to cause privacy problems for both the sender and the recipient. If the adversary assumes common input ownership heuristic for all the inputs, she links the inputs to the previous transactions of the other participant. The sender may encounter a serious problem if she is not knowledgeable about PayJoin technique and the way it is created. In the current implementation of the wallets, the actions that should be done to send the coins as PayJoin is similar to sending the coins as an ordinary payment. Thus, the novice user is involved in a mixing technique, while she might not know the consequence of linking her inputs to the recipient inputs. In the worst case, if the recipient's input is related to the Darknet or criminal activities, the user may get into trouble. If the community seeks a technique that can increase the user's privacy by making the common input ownership heuristic less effective, they should also consider the privacy of the user who gets involved in these transactions. Any side effects as a result of lacking the knowledge by the user should be avoided. On the other hand, PayJoin transactions may render cluster analysis more difficult than before, as it creates false positives in the analysis leading to more super clusters.

In its current state, PayJoin is only described for transactions with more than one input and two outputs. This leads to an anonymity set of 15.4% compared to the total number of transactions according to our extracted data for September 2020 (Figure 2). Moreover, it appears that classical payment transactions have been decreasing over the past years. Therefore, PayJoin should also be extended to transactions with more than two outputs to increase the anonymity set. As one of the main contributions of PayJoin is breaking the common input ownership heuristic, it can also poison the heuristic for a larger set of transactions.

Acknowledgment

This work is supported by COMET SBA-K1 and the Austrian Research Promotion Agency (FFG) via project number 874019.

References

1. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on Internet measurement conference, pp. 127–140. Association for Computing Machinery, New York (2013)
2. Kalodner, H., Möser, M., Lee, K., Goldfeder, S., Plattner, M., Chator, A., and Narayanan, A.: BlockSci: Design and applications of a blockchain analysis platform. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 2721–2738. USENIX, (2020), Change Address Heuristics, `https://citp.github.io/BlockSci/\reference/heuristics/change.html`. Last accessed 20 Sep 2020
3. Improving Privacy Using Pay-to-EndPoint (P2EP), `https://blockstream.com/2018/08/08/en-improving-privacy-using-pay-to-endpoint/`. Last accessed 20 Sep 2020
4. Bustapay BIP: a practical sender/receiver coinjoin protocol, `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2018-August/016340.html`. Last accessed 20 Sep 2020
5. PayJoin, `https://joinmarket.me/blog/blog/payjoin/`. Last accessed 23 Aug 2020
6. BIP78: A Simple Payjoin Proposal, `https://github.com/bitcoin/bips/blob/master/bip-0078.mediawiki`. Last accessed 20 Sep 2020
7. BitCoin Privacy, `https://en.bitcoin.it/wiki/Privacy`. Last accessed 20 Sep 2020
8. Bitcoin privacy (Payjoin/P2EP), `https://diyhpl.us/wiki/transcripts/london-bitcoin-devs/2020-05-05-socratic-seminar-payjoins/`. Last accessed 20 Sep 2020