Knapsack CoinJoin

ANONYMOUS COINJOIN TRANSACTIONS WITH ARBITRARY VALUES

Anonymous CoinJoin Transactions with Arbitrary Values (2017)

Anonymous CoinJoin Transactions with Arbitrary Values

Felix Konstantin Maurer
Communication and Distributed Systems
RWTH Aachen University
felix.maurer@comsys.rwth-aachen.de

Till Neudecker
Institute of Telematics
Karlsruhe Institute of Technology
till.neudecker@kit.edu

Martin Florian
Institute of Telematics
Karlsruhe Institute of Technology
florian@kit.edu

https://www.comsys.rwth-aachen.de/fileadmin/papers/2017/2017-maurer-trustcom-coinjoin.pdf

Knapsack Coinjoin Anonymity Analysis (2019), Lucas Ontivero (PDF Available upon request)

The privacy problem with Bitcoin

- Transactions are public
- Transactions point to previous transactions
- Transactions (can) leak future spending

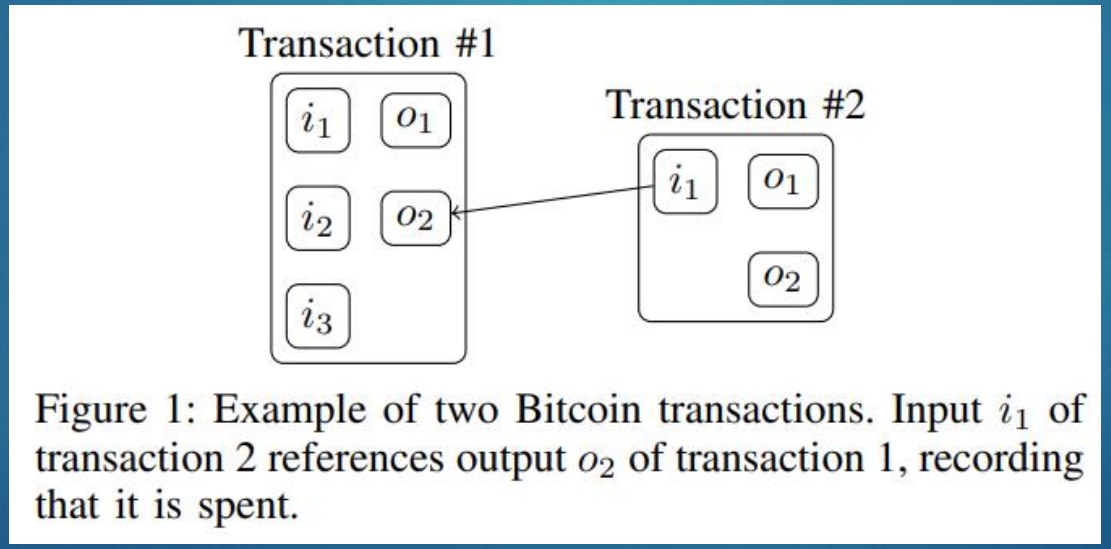


Figure 1: Example of two Bitcoin transactions. Input i_1 of transaction 2 references output o_2 of transaction 1, recording that it is spent.

Idea #1 – Joining Transactions

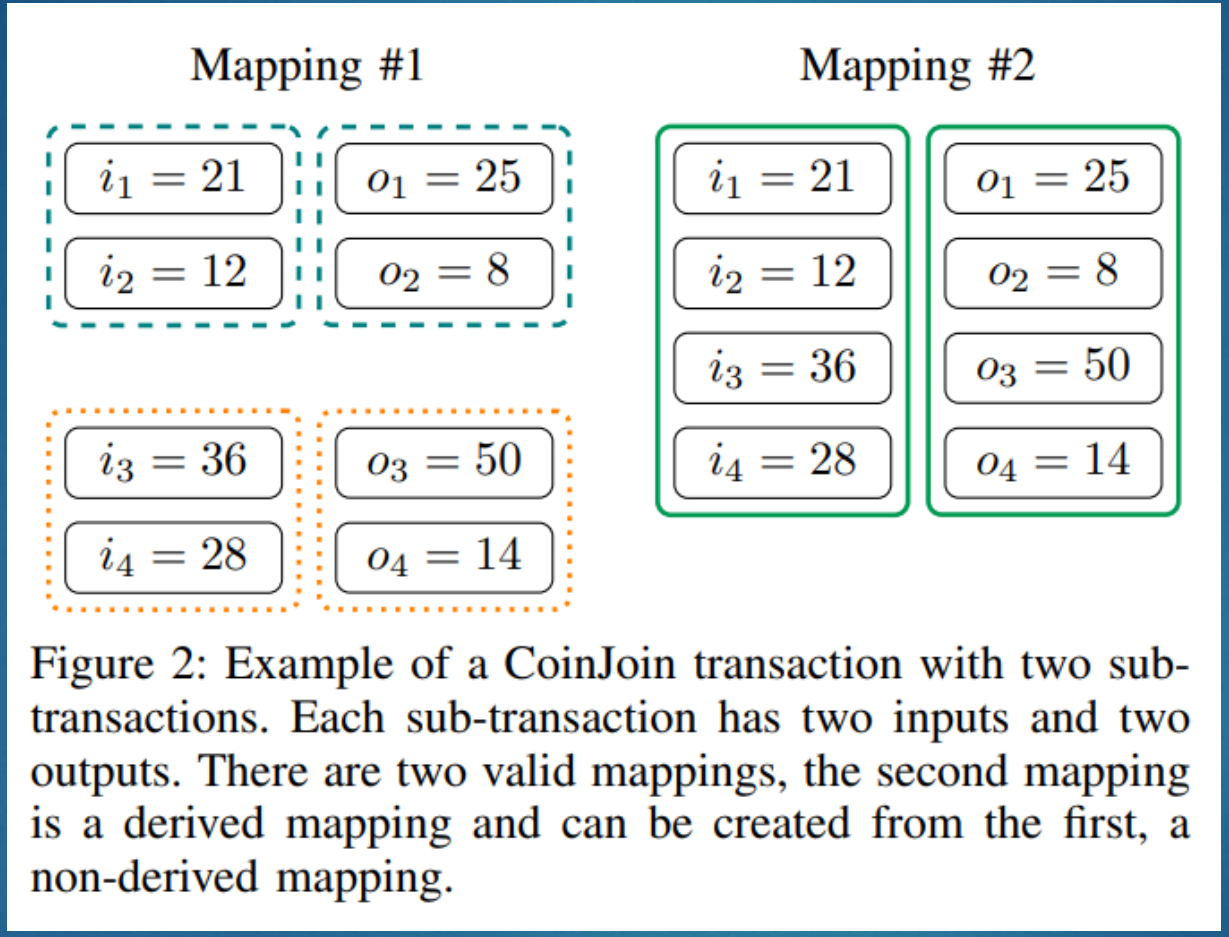
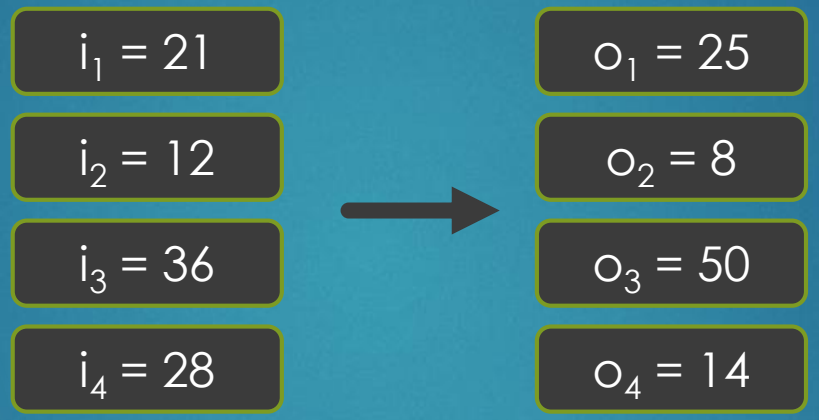


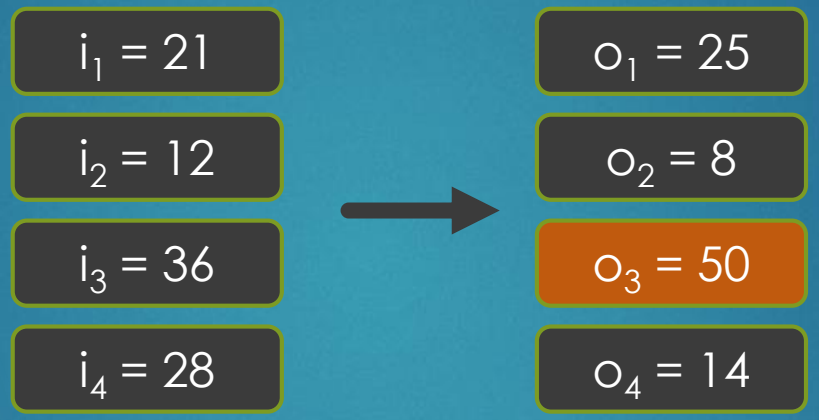
Figure 2: Example of a CoinJoin transaction with two sub-transactions. Each sub-transaction has two inputs and two outputs. There are two valid mappings, the second mapping is a derived mapping and can be created from the first, a non-derived mapping.

Idea #1 – Joining Transactions

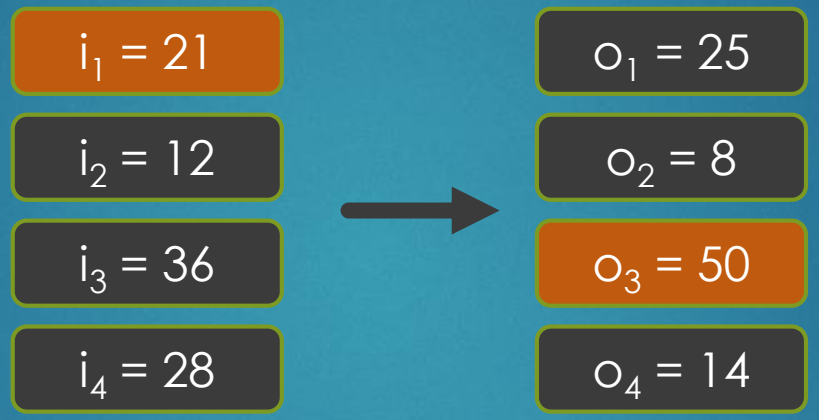
Idea #1 – Joining Transactions



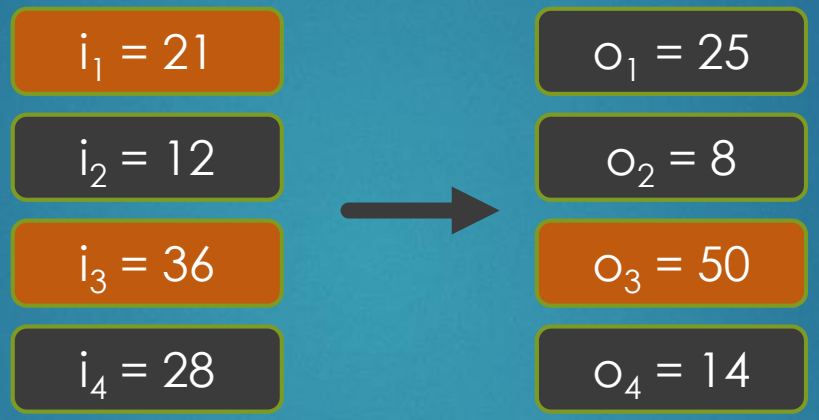
Idea #1 – Joining Transactions



Idea #1 – Joining Transactions

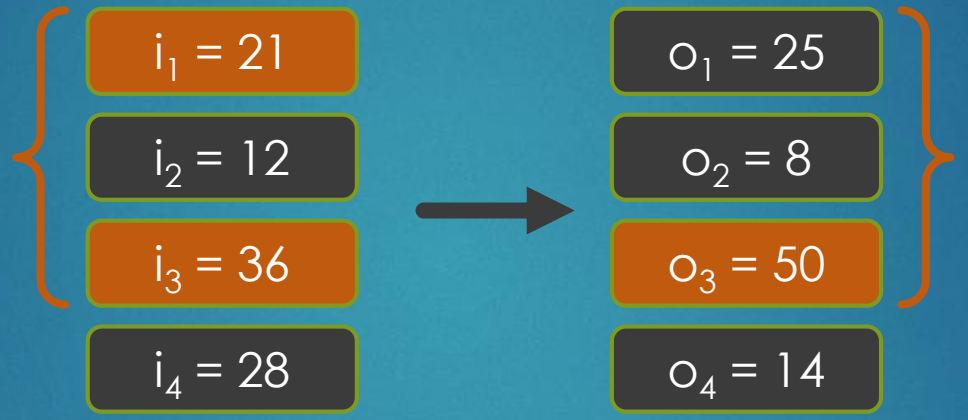


Idea #1 – Joining Transactions



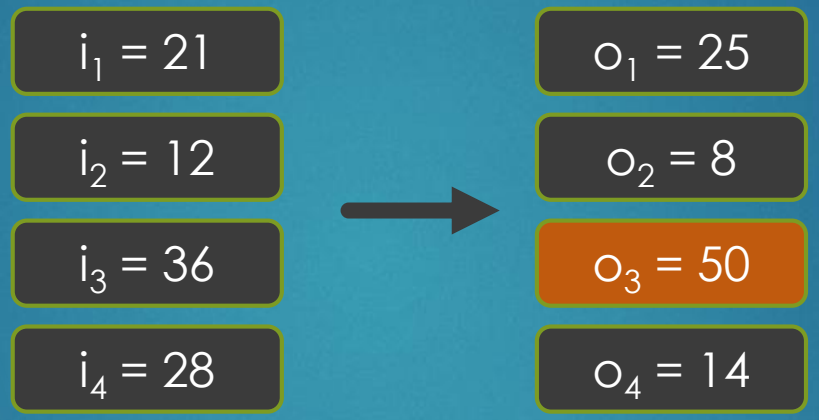
Idea #1 – Joining Transactions

$$\sum_{k=0}^{n} i_i = 57$$

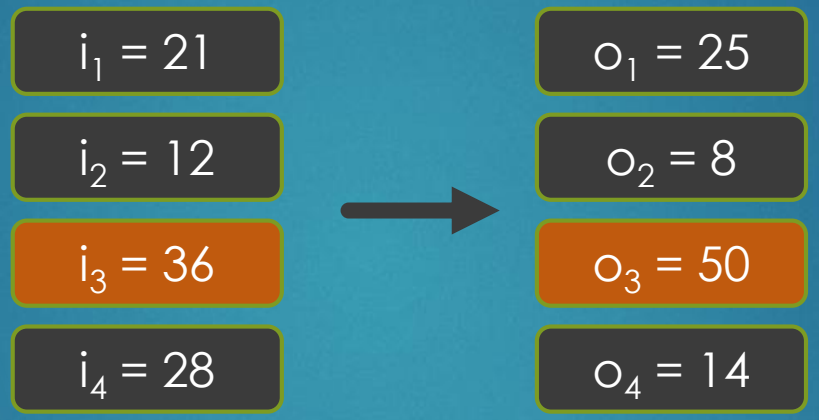


$$\sum_{k=0}^{n} o_i = 50$$

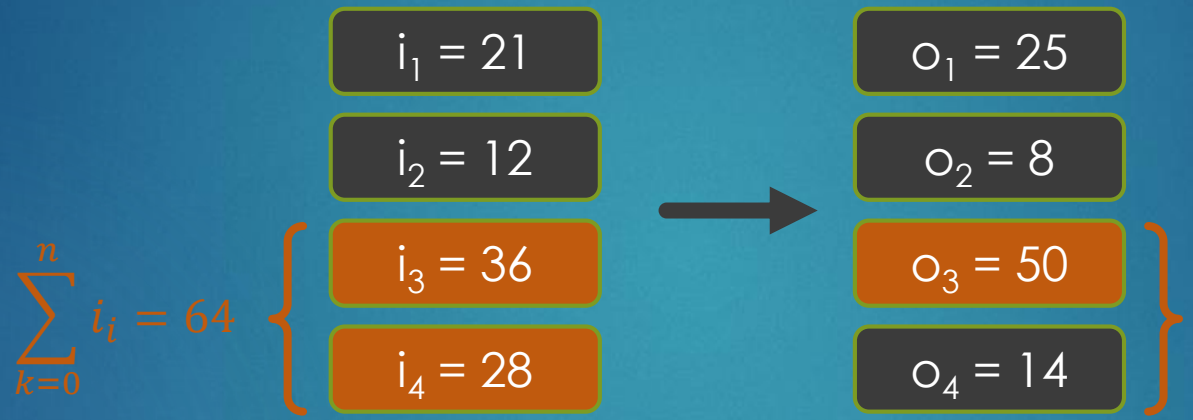
Idea #1 – Joining Transactions



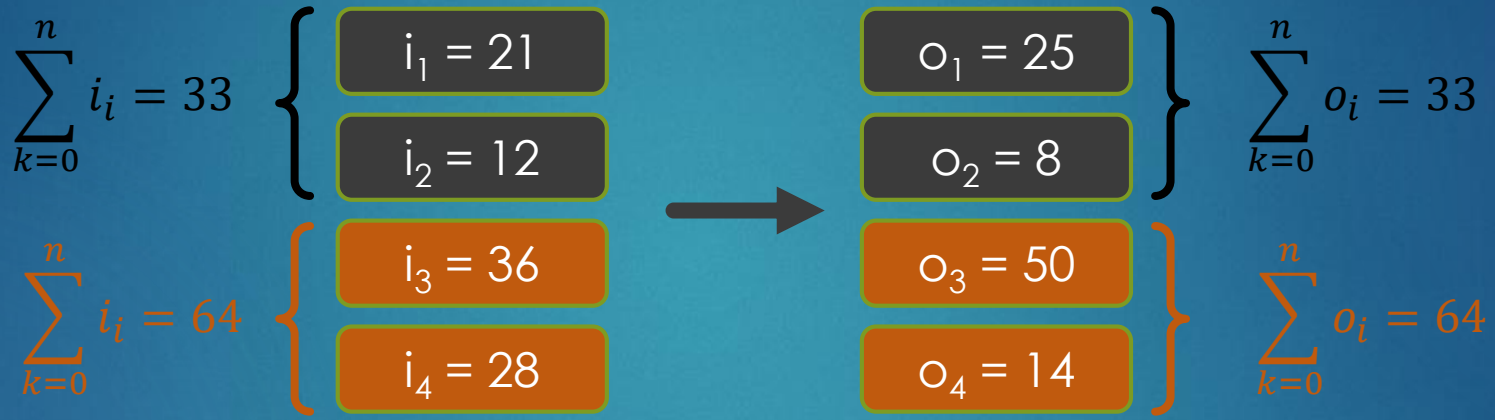
Idea #1 – Joining Transactions



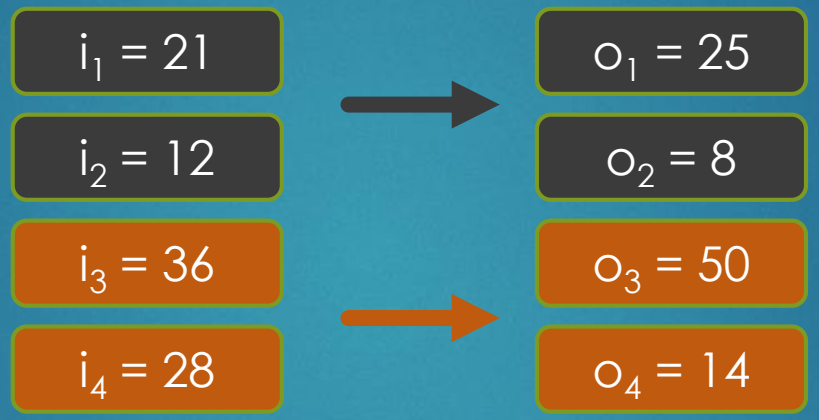
Idea #1 – Joining Transactions



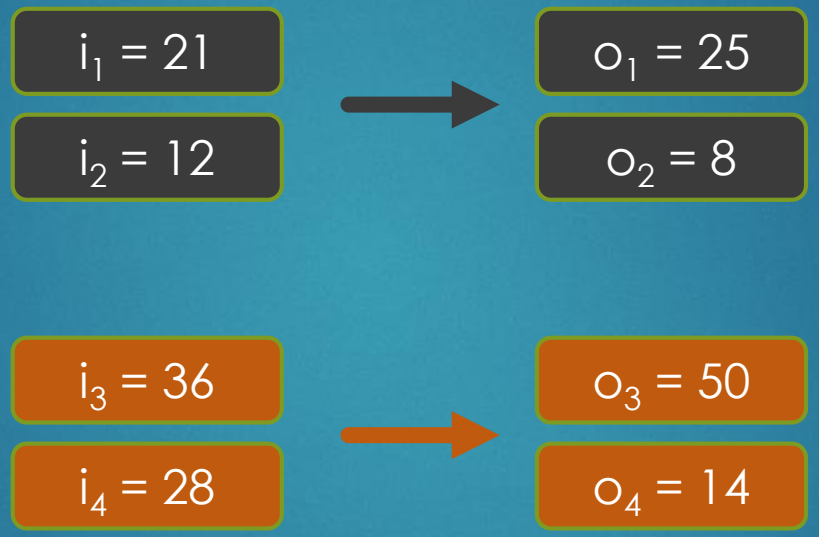
Idea #1 – Joining Transactions



Idea #1 – Joining Transactions



Idea #1 – Joining Transactions



Idea #1 – Joining Transactions

- How good is the anonymity provided by this simple model?
- Can we improve upon it?

Formal Definitions

1. Coinjoin Transaction
2. Inputs
3. Outputs
4. Coins

$T = (I, O, v)$

$\mathrm{I} = \{i_1, \ldots, i_m\}$

$\mathrm{O} = \{o_1, \ldots, o_m\}$

$C = I \cup O$

$$\sum_{i \in I} v(i) = \sum_{o \in O} v(o)$$

Formal Definitions

A CoinJoin transaction consists of sub-transactions tk, … **there must be at least one way of partitioning all inputs and outputs, so that each subset of inputs has exactly one corresponding subset of outputs with which it forms a sub-transaction.** We call this a mapping, as it maps inputs to outputs.

Formal Definitions

5. Set of all partitions $\Phi(S)$
6. Set of all Input Partitions $\boldsymbol{O} \in \Phi(O)$
7. Set of all output Partitions $\boldsymbol{I} \in \Phi(I)$
8. Set of all mappings M

$$\mathrm{M} = \left\{ m: \boldsymbol{I} \to \boldsymbol{O}\ s.t. \forall I \in \boldsymbol{I}, \sum_{i \in I} v(i) = \sum_{o \in m(I)} v(o) \right\}$$

In simpler terms

We want to be able to break a transaction down into ***subtransactions*** which are combinations of inputs and outputs that add up to the same value. A subtransaction that consists of smaller subtransactions is called a ***derived*** subtransaction. We will concern ourselves with ***non-derived*** sub transactions.

Anonymity in this framework

The probability that an input coin and an output coin belong to the same transaction are as follows. It is defined as the number of mappings where the input coin is in a subset of the input partition that is assigned to the subset of the output partition of which the output coin is an element, divided by the total number of mappings.

Anonymity in this framework

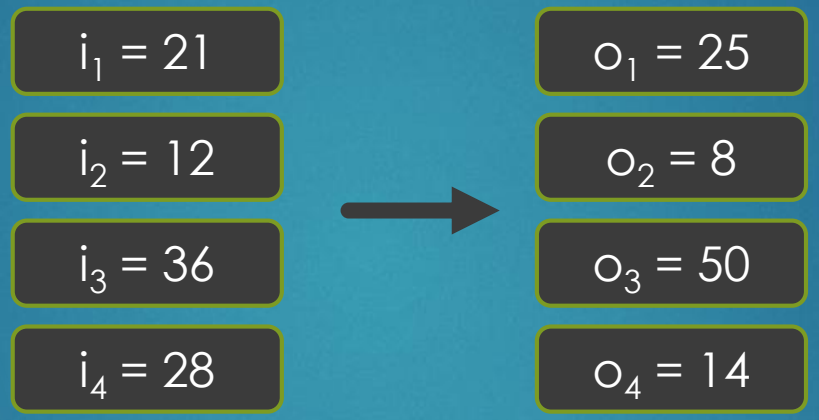
$$i_1, i_2 \in I, \quad p_{II}(i_1, i_2) = \frac{|\{M = (\mathcal{I}, \mathcal{O}, m) \in \mathcal{M} \mid \exists \mathtt{I} \in \mathcal{I} : i_1, i_2 \in \mathtt{I}\}|}{|\mathcal{M}|}$$

The probability p_{OO} that two outputs belong to the same sub-transaction can be defined correspondingly.

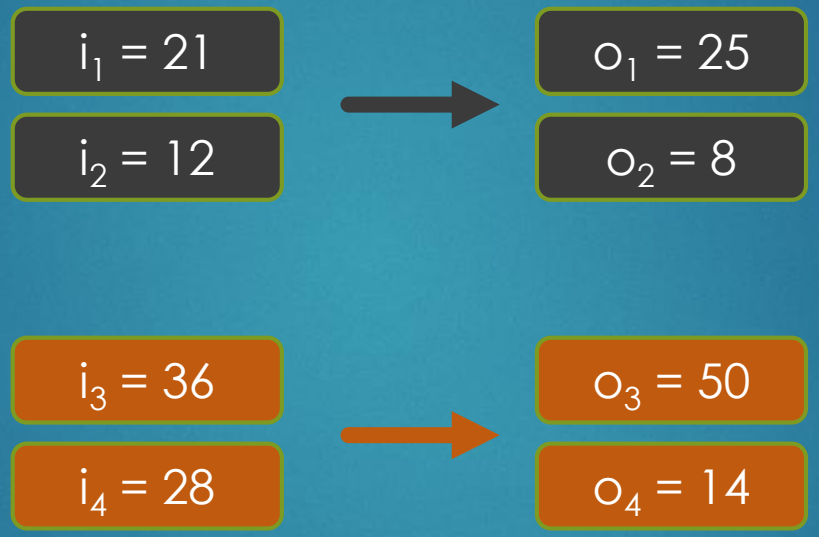
We define the probability $p_{IO}(i, o)$ that an input coin and an output coin belong to the same transaction as follows. It is defined as the number of mappings where the input i is in a subset of the input partition that is assigned to the subset of the output partition of which o is an element, divided by the total number of mappings.

$$i \in I, o \in O, \quad p_{IO}(i, o) = \frac{|\{M = (\mathcal{I}, \mathcal{O}, m) \in \mathcal{M} \mid \exists \mathtt{I} \in \mathcal{I} : i \in \mathtt{I} \wedge o \in m(\mathtt{I})\}|}{|\mathcal{M}|}$$

Idea #1 – Joining Transactions



Idea #1 – Joining Transactions



Evaluation of Joined Transactions

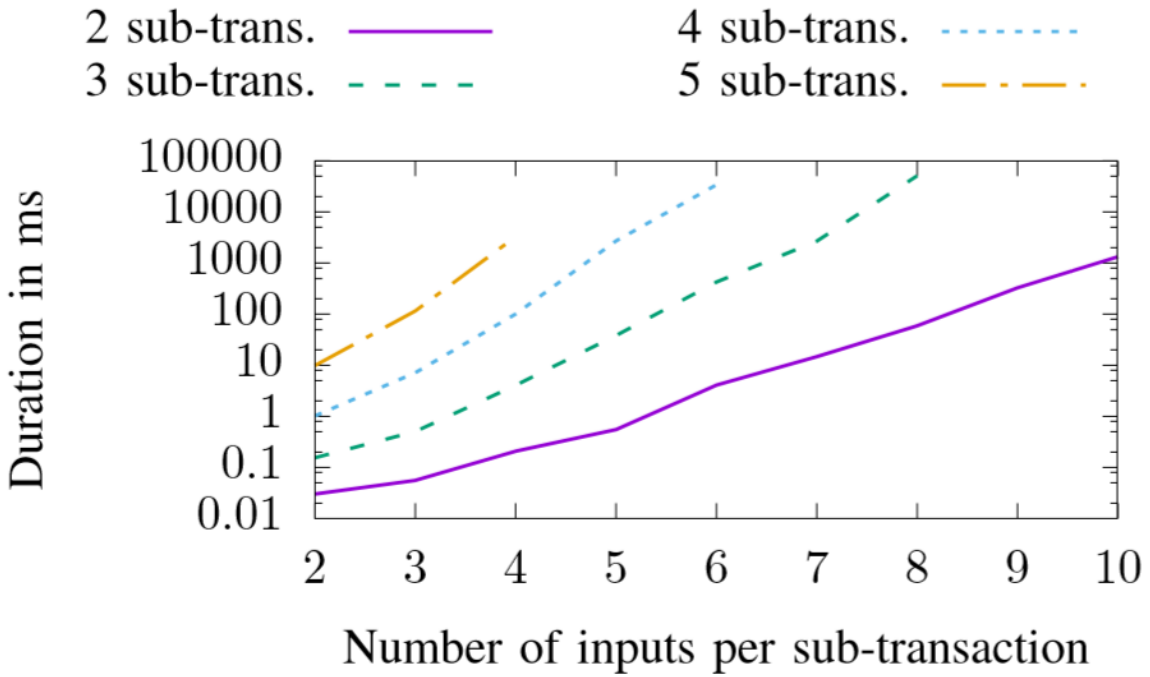


Figure 3: The processing time for calculating all mappings for a CoinJoin transaction. For each number of sub-transactions there is one line and on the x-axis the number of inputs per sub-transaction are marked.

Processing Time

Evaluation of Joined Transactions

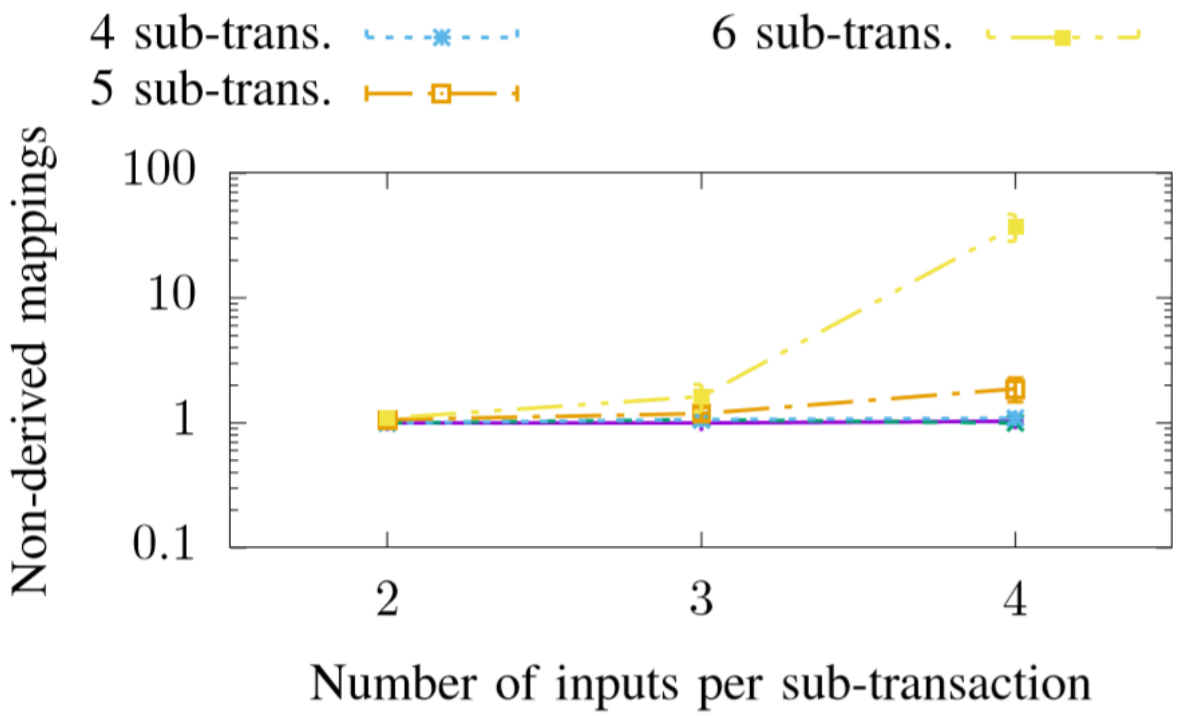
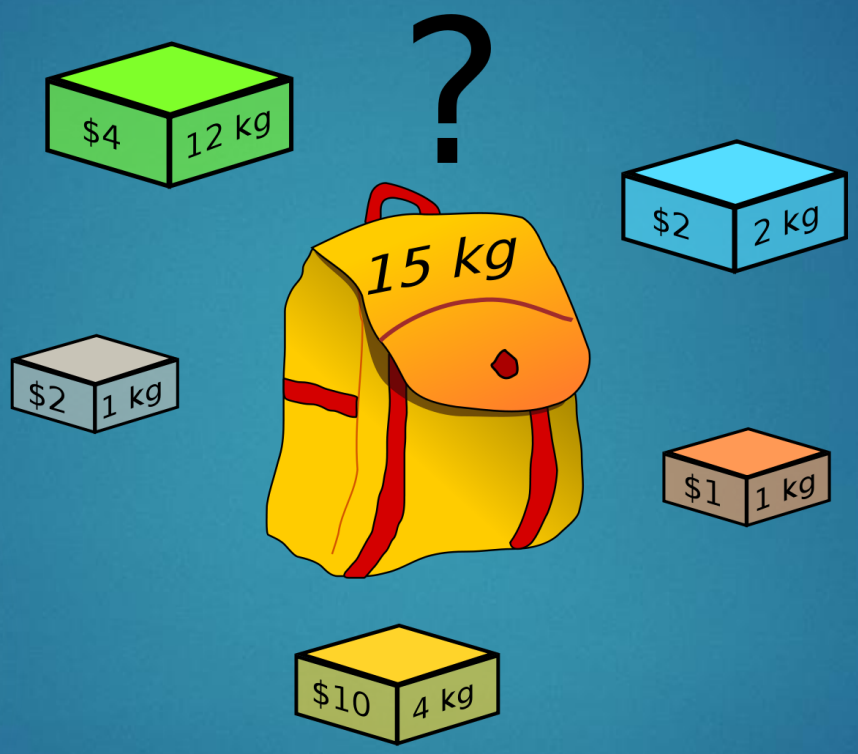


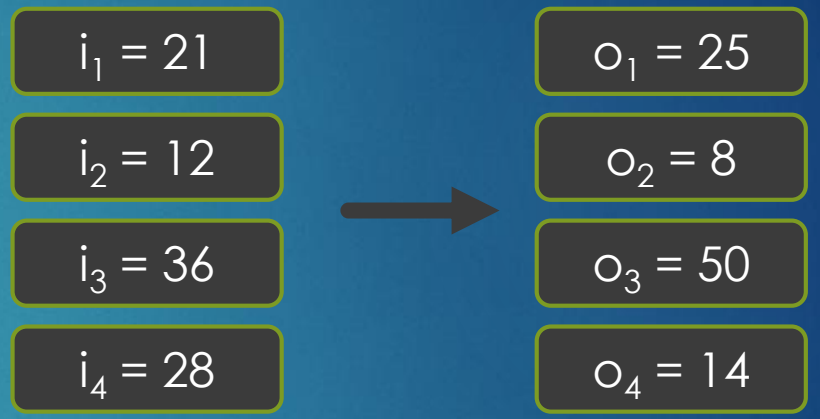
Figure 5: The number of non-derived mappings for CoinJoin transactions that have not been mixed.

Non-Derived Mappings

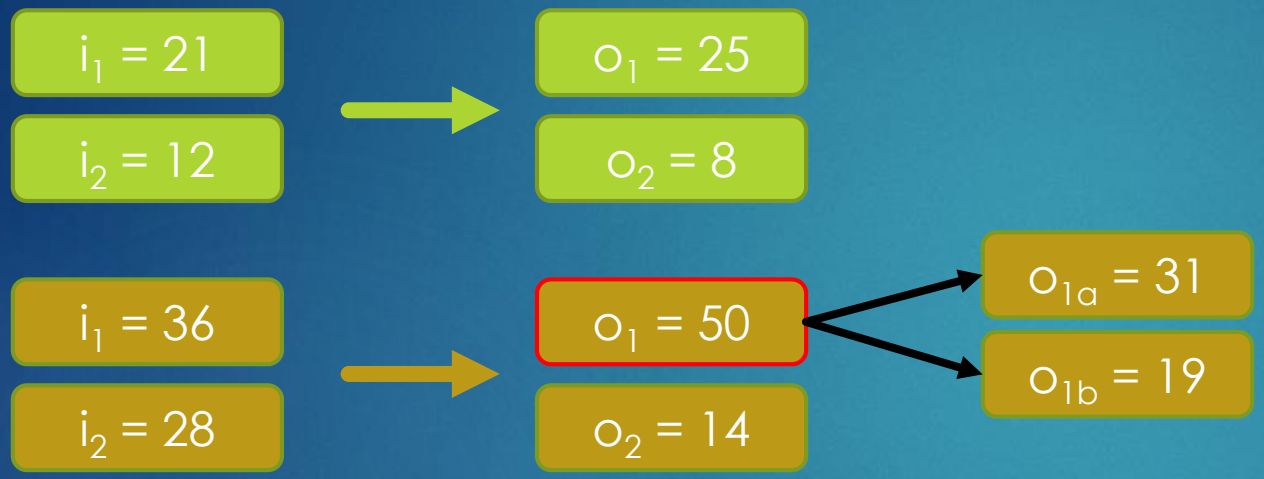
Idea #2 – Knapsack Mixing



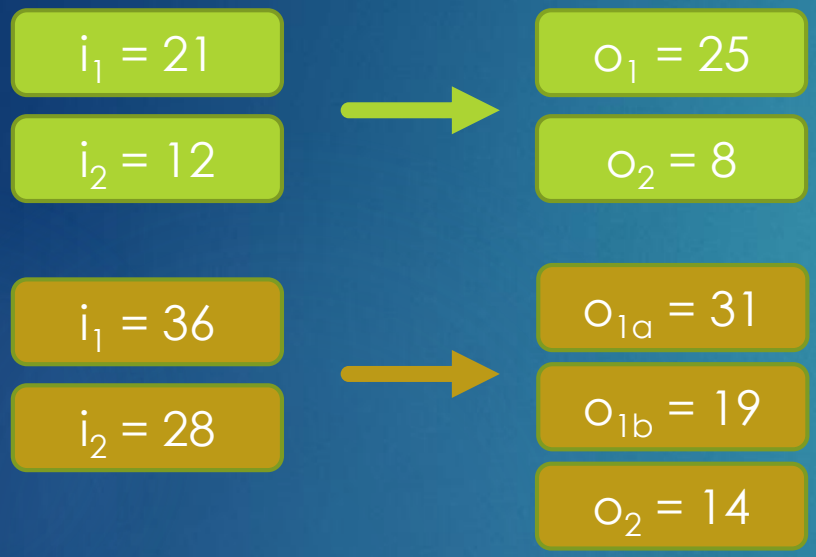
Idea #2 – Knapsack Mixing

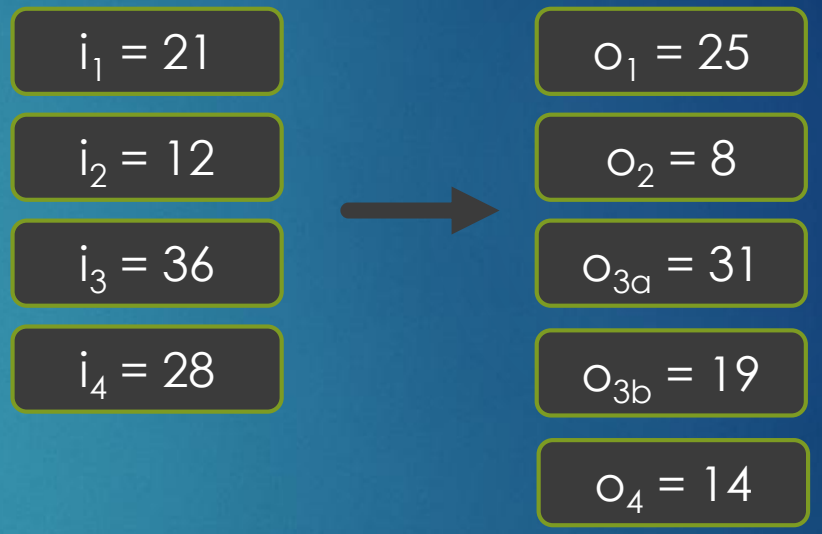


Idea #2 – Knapsack Mixing

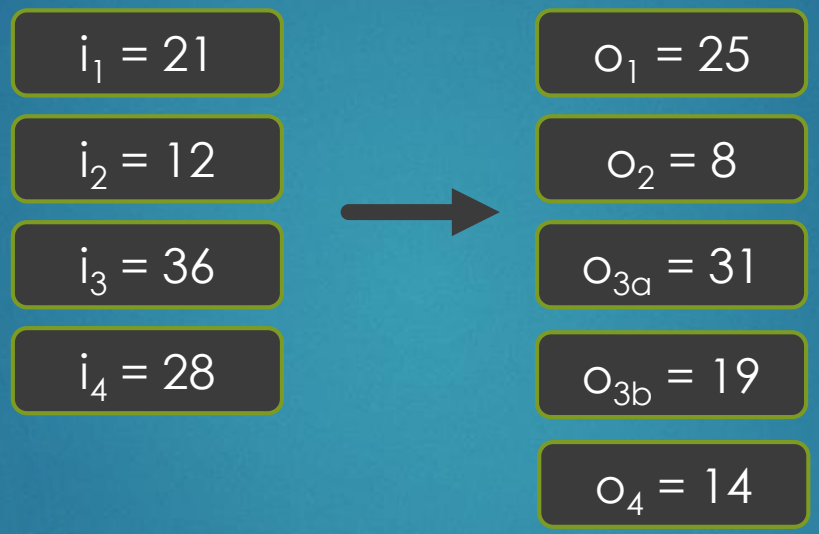


Idea #2 – Knapsack Mixing

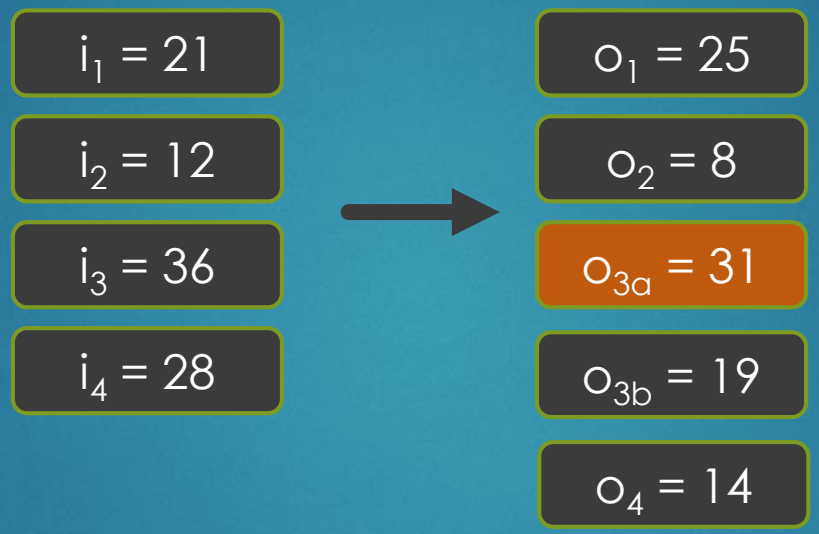




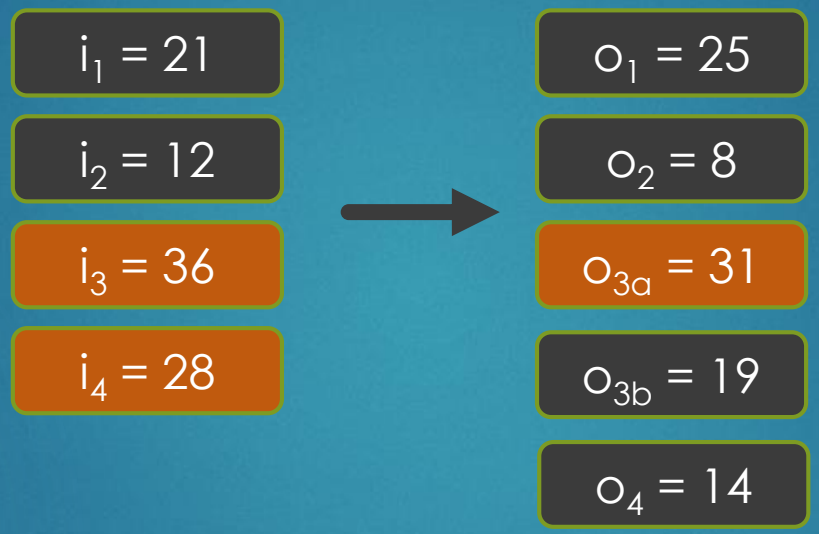
Idea #2 – Knapsack Mixing



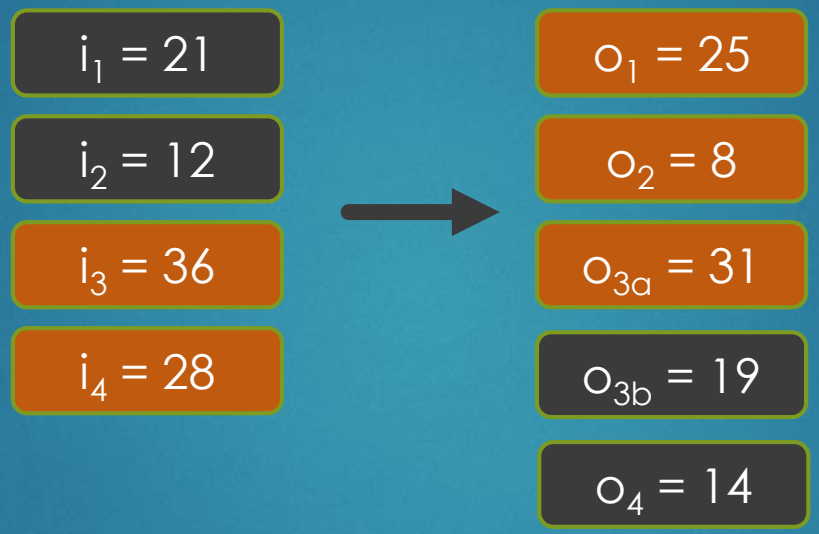
Idea #2 – Knapsack Mixing



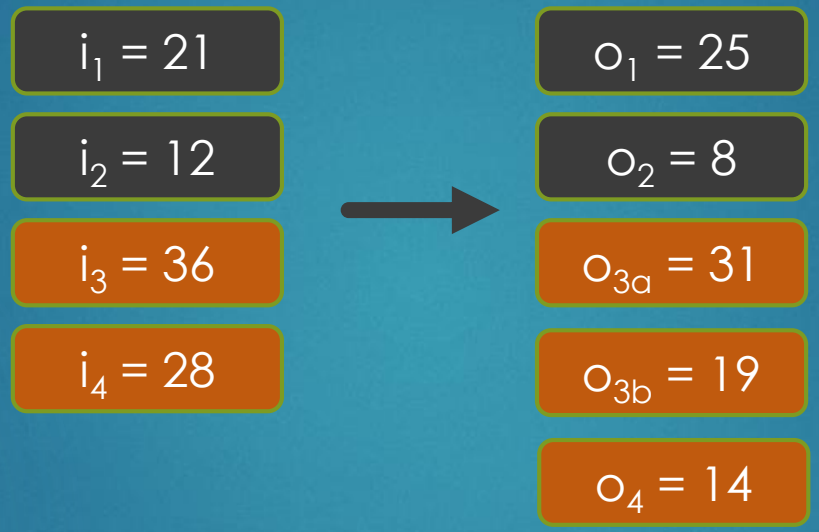
Idea #2 – Knapsack Mixing



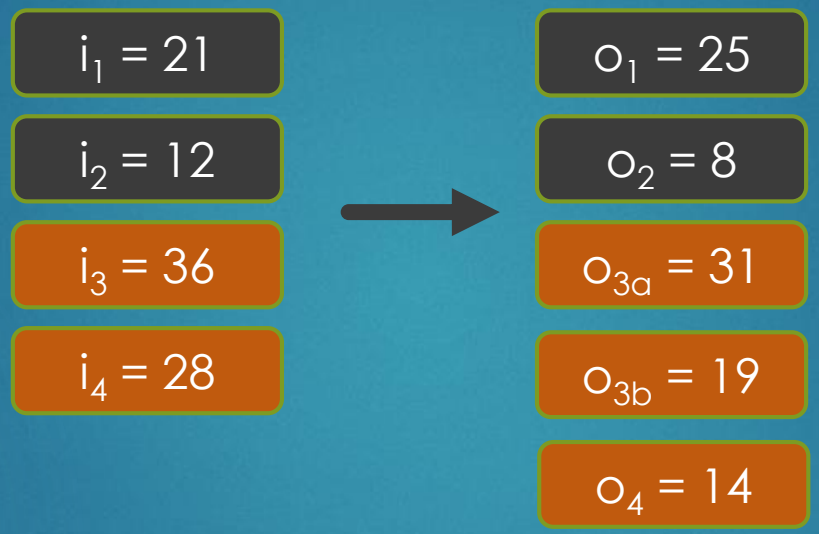
Idea #2 – Knapsack Mixing



Idea #2 – Knapsack Mixing



Idea #2 – Knapsack Mixing**



Evaluation

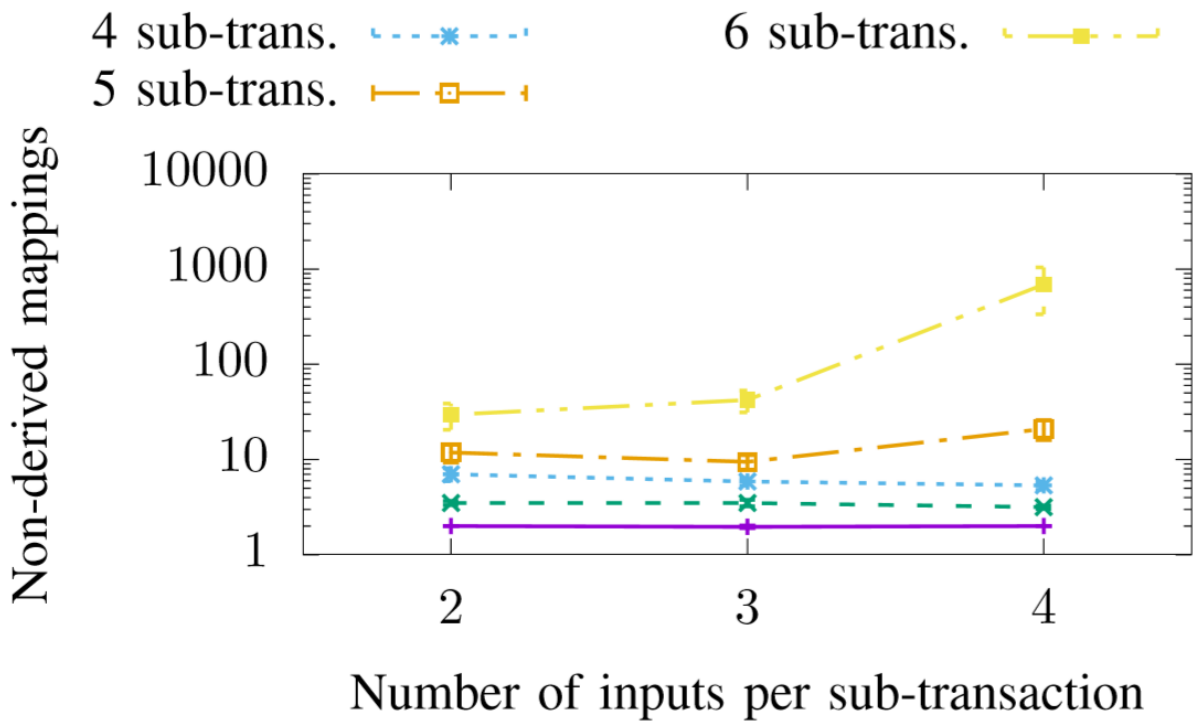


Figure 7: The number of non-derived mappings for CoinJoin transactions that have been mixed with our simple input shuffling algorithm.

Evaluation

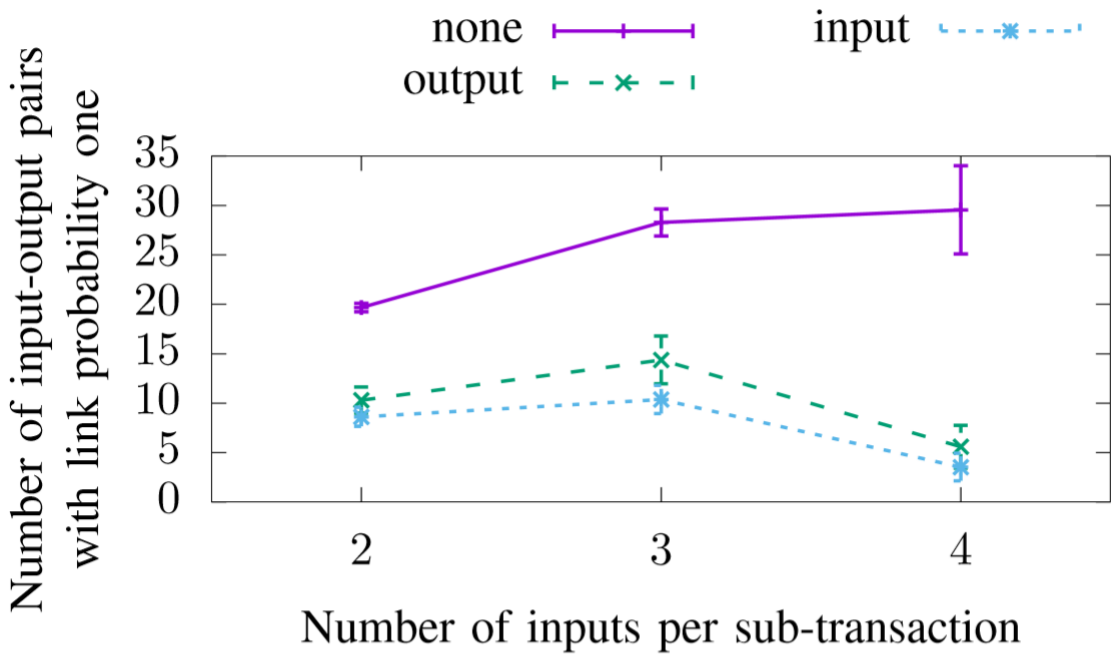


Figure 8: The number of input-output-pairs with link probability $p_{IO}(i, o) = 1$ in CoinJoin transactions with 5 sub-transactions, depending on the mixing algorithm used.

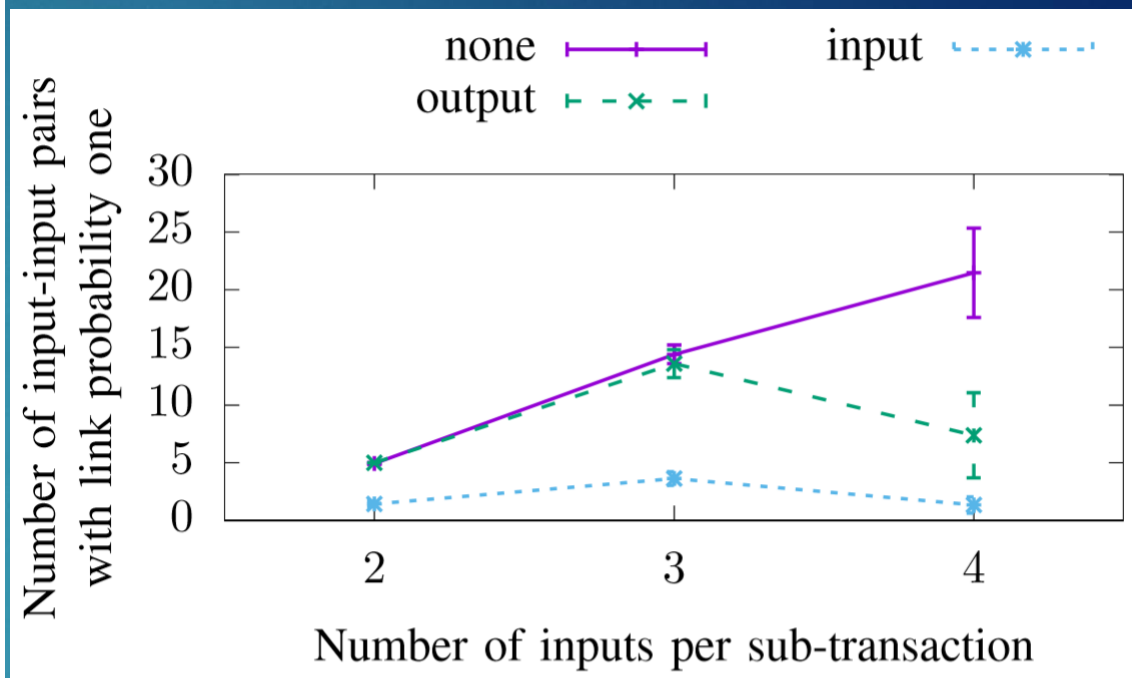


Figure 9: The number of input-input-pairs with link probability $p_{II}(i_1, i_2) = 1$ in CoinJoin transactions with 5 sub-transactions, depending on the mixing algorithm used.

Concluding Thoughts

- Anonymity in ***Computational Complexity***
- Link between inputs, outputs or both still exist
- The entire sub-transaction partition must be known in order to apply the algorithm
- Upper bound anonymity is still current ZeroLink
- Algorithm relies on increasing number of inputs and outputs (costly)

Now let's talk ideas

- Wasabi should apply Knapsack against the change in ZeroLink mixes.
- We should think deeply about Wasabi as a mixing service vs. a sending service
- To what extent can we improve trust-lessness?

Knapsack CoinJoin

ANONYMOUS COINJOIN TRANSACTIONS WITH ARBITRARY VALUES