# CashFusion

FLEXIBLE ARBITRARY-INPUT CONSOLIDATION COINJOINS**

# Fyookball and Lundeberg (2019)

## CashFusion

Authors: Jonald Fyookball, Mark B. Lundeberg

### Introduction

**THE PROBLEM:**

CashShuffle is a powerful tool for obfuscating the origin of a coin. However, after shuffling a wallet, a user will inevitably wish to consolidate several coins, and for this another tool is needed. We need a method to coordinate coinjoin transactions with multiple inputs per user. This is inherently challenging because we want to hide input linkages while simultaneously attempting to blame/ban users who don't sign all their inputs.

https://github.com/cashshuffle/spec/blob/master/CASHFUSION.md

# Wasabi Research Club

- January 6th, 2020 – Knapsack CoinJoin
- January 13th, 2020 – SNICKER
- January 20th, 2020 – CoinShuffle
- January 27th, 2020 – Dining Cryptographer Networks
- February 3rd, 2020 – CoinShuffle ++ (Part 1)
- Feburary 10th, 2020 – CoinShuffle ++ (Part 2, w/ Tim Ruffing)
- Feburary 17th, 2020 – CashFusion
- February 24th, 2020 – TBD

https://github.com/zkSNACKs/WasabiResearchClub

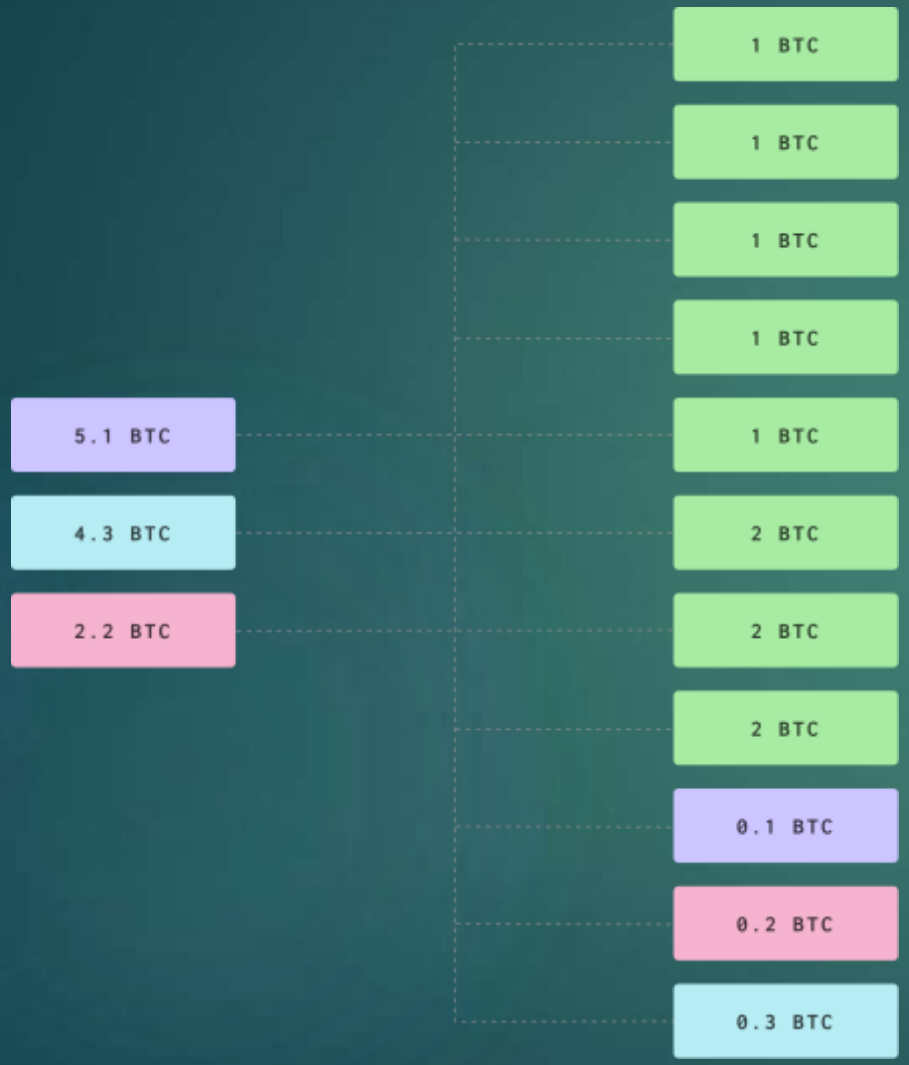
# The last three weeks- CoinShuffle(++)

- One issue with CoinJoin implementations is the reliance on a central coordinator**. Removing the coordinator** would require a secure method of participants declaring their anonymous addresses
- We can replace the coordinator with a Coin*Shuffle*, where each participant **onion-encrypts their address** with the public keys of the latter participants. They then **decrypt and shuffle** all encrypted addresses they have received with their own address, and proceed to hand off the encrypted addresses to the next participant.
- **Scales poorly** with many participants, ElectronCash(5)
- *CoinShuffle*++ adds to the protocol by introducing a DC-nets protocol that can handle collisions and disruptions in just ***4+2f*** rounds, given ***f*** peers.

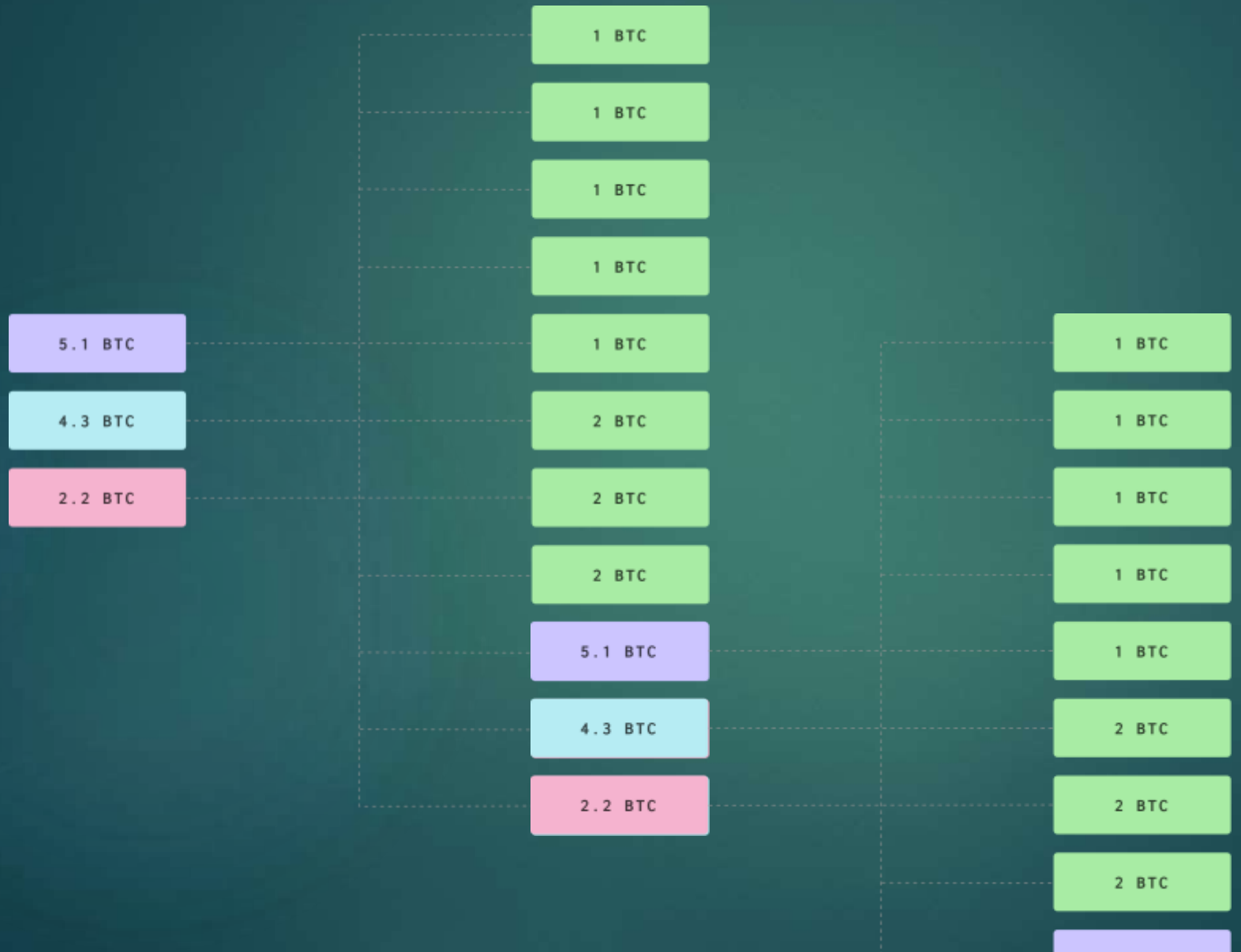
# The problem(s)

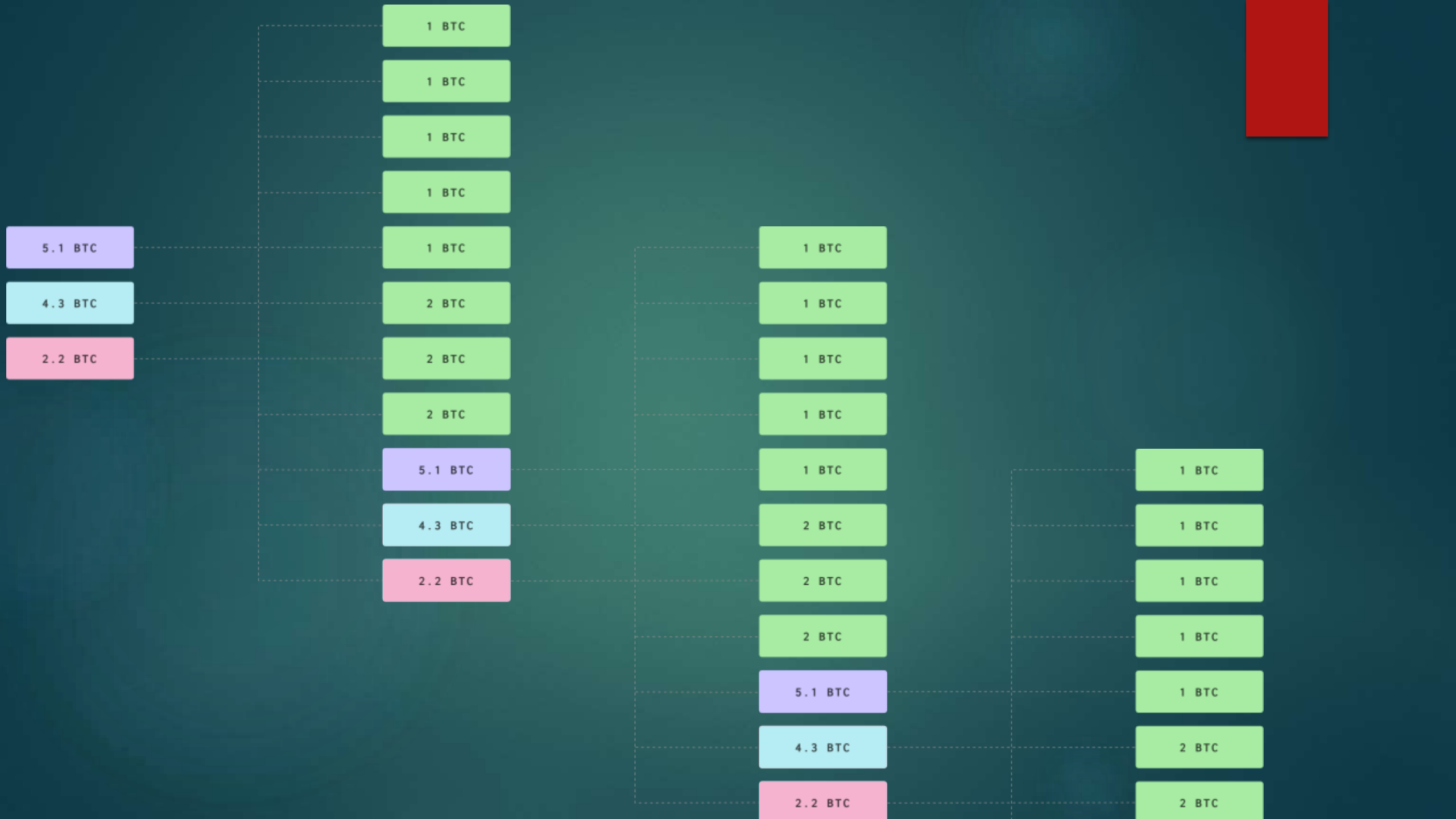
- When users CoinJoin with CashShuffle (or any protocol for that matter) they typically get many coins of equal denominations.
- We need a **private way** for users to **consolidate arbitrary number of coins** without revealing input ownership.
- The protocol should not rely on a central coordinator.
- The protocol should allow **for arbitrary outputs of arbitrary amounts**.

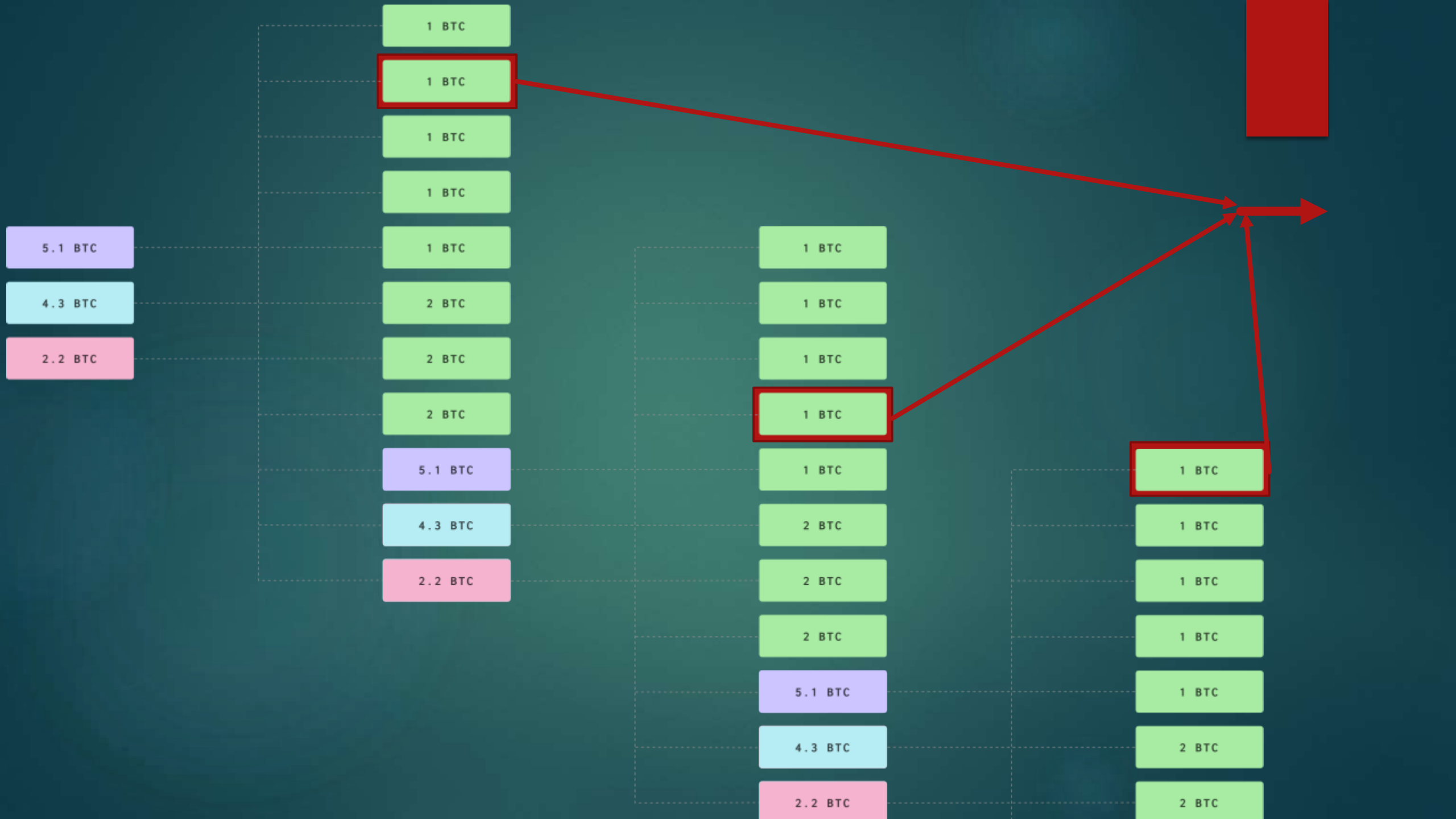
# The problem(s)



# The problem(s)




1 BTC
1 BTC
1 BTC
1 BTC
5.1 BTC
1 BTC
4.3 BTC
2 BTC
2.2 BTC
2 BTC
2 BTC
5.1 BTC
4.3 BTC
2.2 BTC
1 BTC
1 BTC
1 BTC
1 BTC
2 BTC
2 BTC
2 BTC
5.1 BTC
4.3 BTC
2.2 BTC
1 BTC
1 BTC
1 BTC
1 BTC
2 BTC
2 BTC


1 BTC
1 BTC
1 BTC
1 BTC
5.1 BTC
1 BTC
4.3 BTC
2 BTC
2.2 BTC
2 BTC
2 BTC
5.1 BTC
4.3 BTC
2.2 BTC
1 BTC
1 BTC
1 BTC
1 BTC
1 BTC
2 BTC
2 BTC
2 BTC
5.1 BTC
4.3 BTC
2.2 BTC
1 BTC
1 BTC
1 BTC
1 BTC
2 BTC
2 BTC

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \ldots, CJ_{n-2}CJ_{n-1}, CJ_n$$

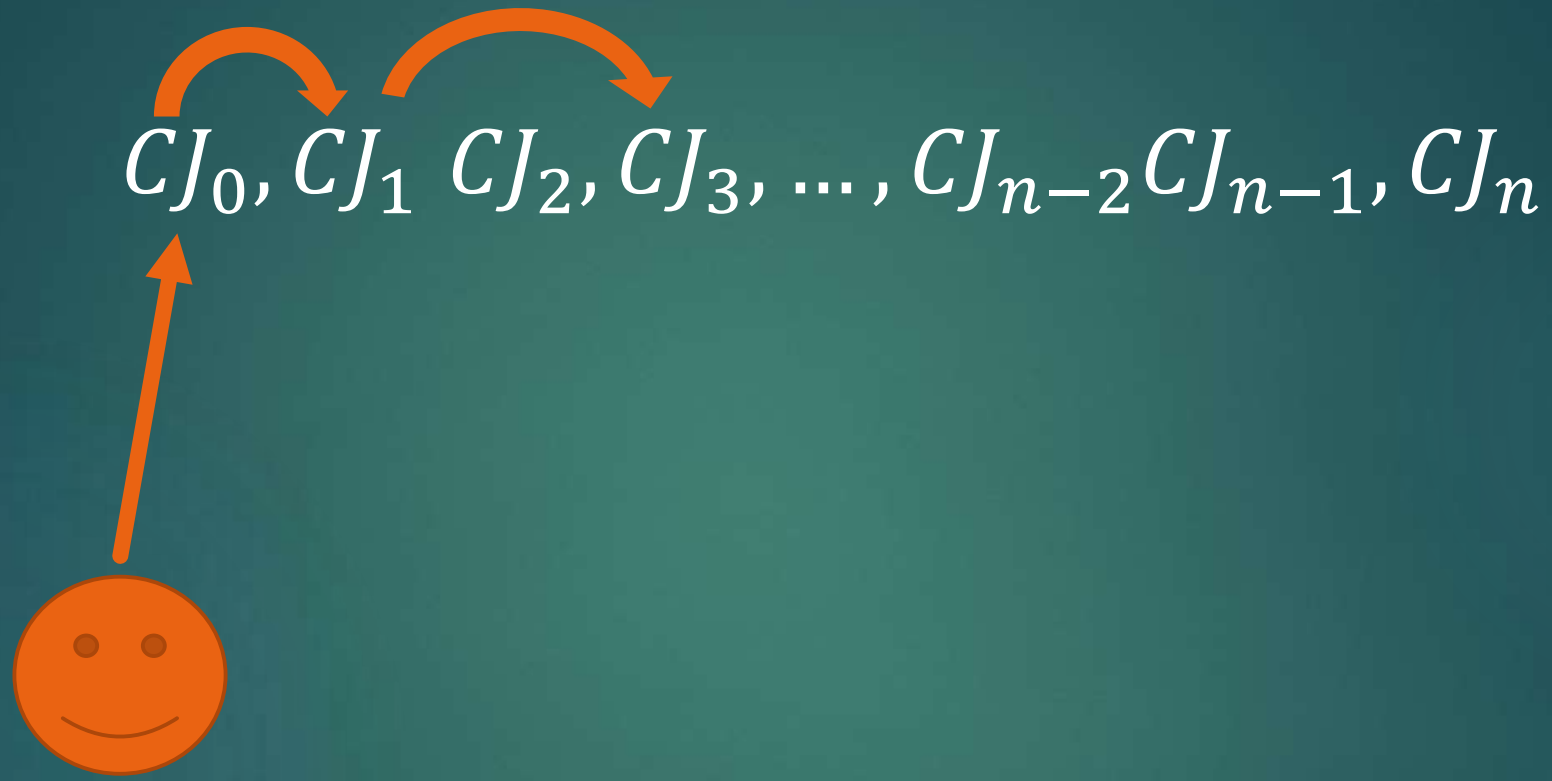
# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \ldots, CJ_{n-2} CJ_{n-1}, CJ_n$$

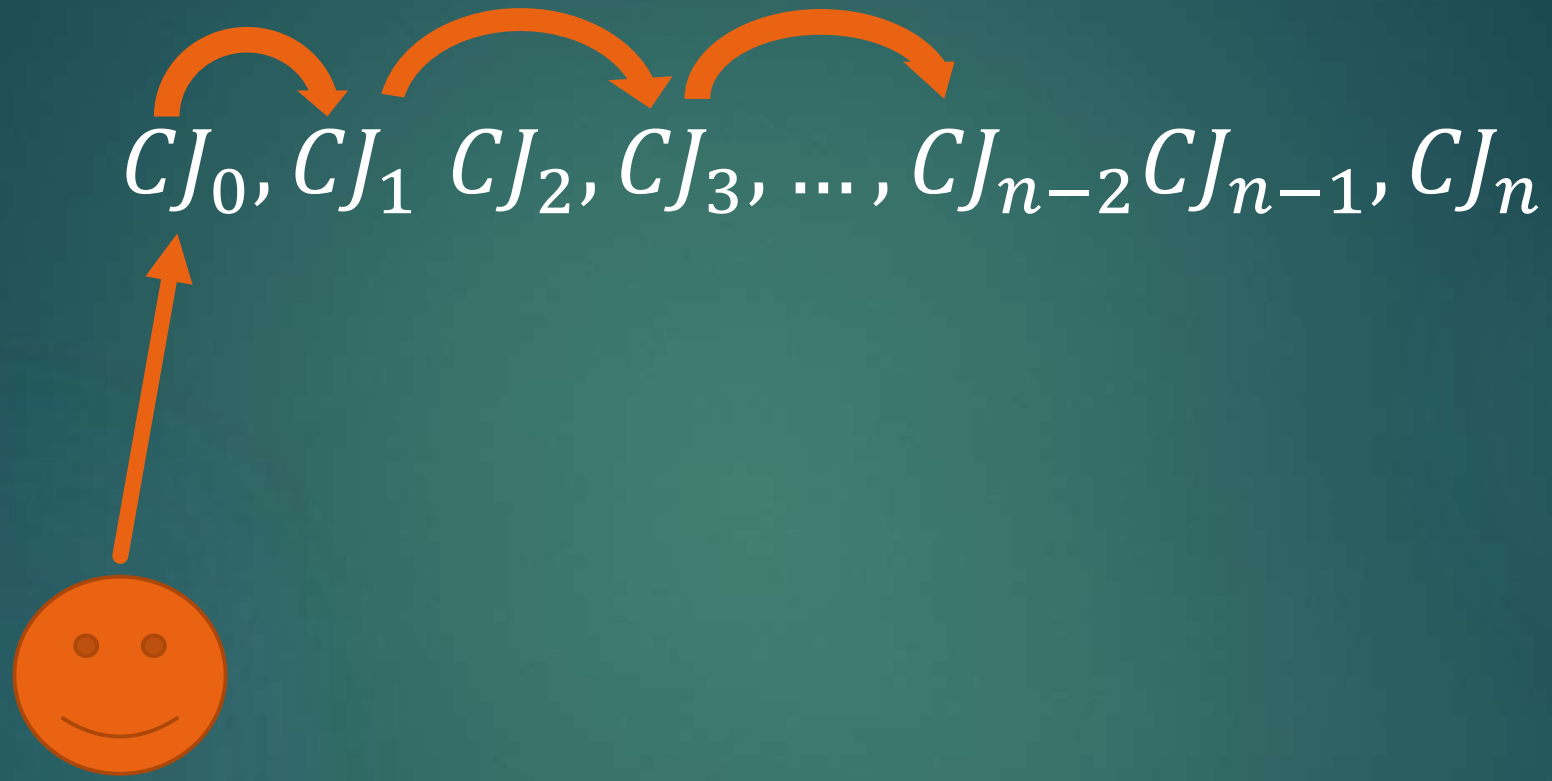
# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \ldots, CJ_{n-2} CJ_{n-1}, CJ_n$$

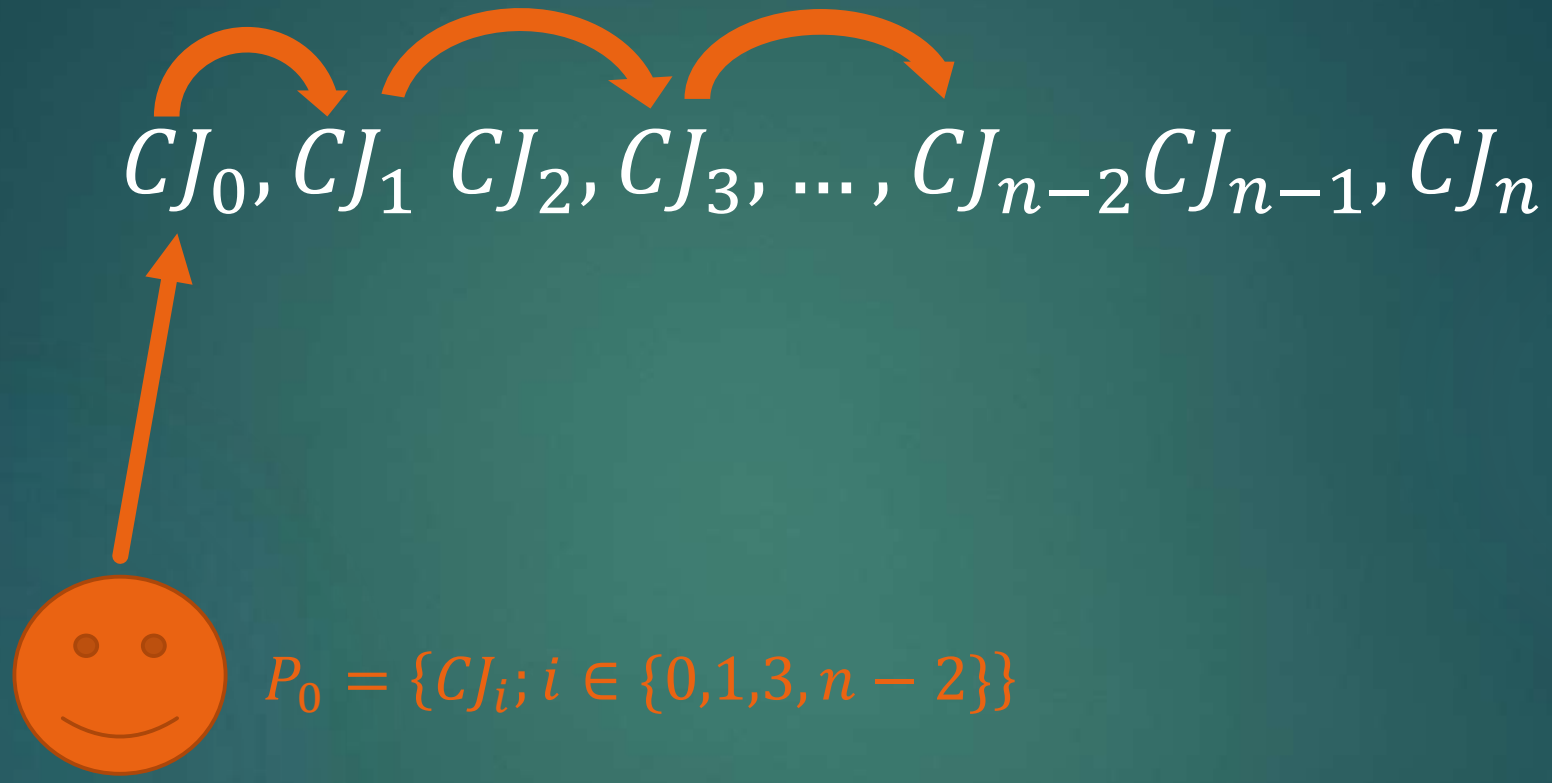
# The problem(s)



# The problem(s)

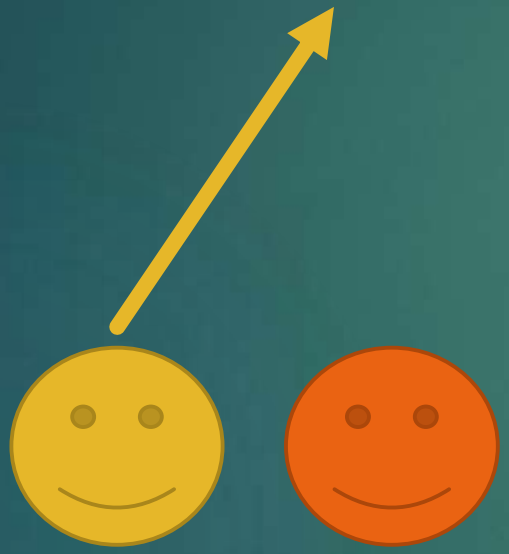


# The problem(s)



# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$



$$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$$

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$

$$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$$

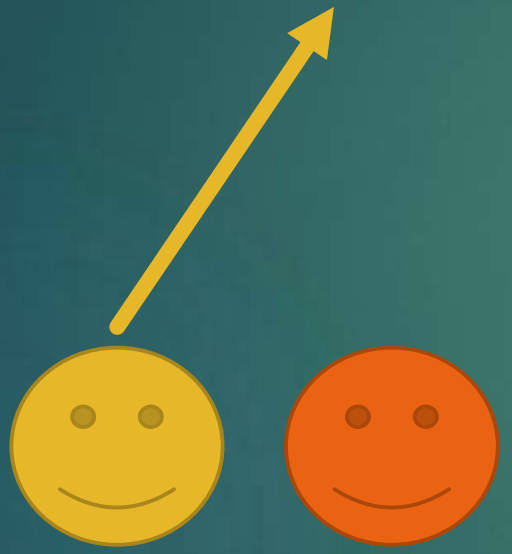
# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$

$$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$$

# The problem(s)

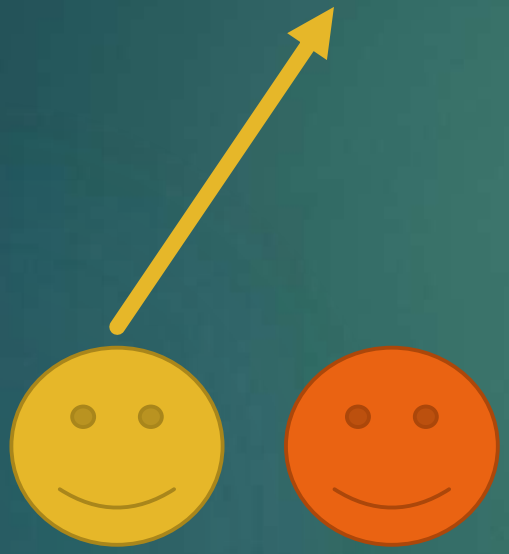
$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2}\ CJ_{n-1}, CJ_n$$



$$P_0 = \{CJ_i; i \in \{0,1,3,n-2\}\}$$

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2}\ CJ_{n-1}, CJ_n$$



$$P_0 = \{CJ_i; i \in \{0,1,3,n-2\}\}$$

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2}\ CJ_{n-1}, CJ_n$$

$$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$$

$$P_1 = \{CJ_i; i \in \{1,2,3, n-1, n\}\}$$

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$

$$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$$

$$P_1 = \{CJ_i; i \in \{1,2,3, n-1, n\}\}$$

$$P_2 = \{CJ_i; i \in \{1, 3, n-2, n\}\}$$

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$

$$I = \{CJ_i; i \in \{2,3, n-1\}\} \longrightarrow O$$

$$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$$

$$P_1 = \{CJ_i; i \in \{1,2,3, n-1, n\}\}$$

$$P_2 = \{CJ_i; i \in \{1, 3, n-2, n\}\}$$

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$

$$I = \{CJ_i; i \in \{2,3, n-1\}\} \longrightarrow O$$

$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$ ✗

$P_1 = \{CJ_i; i \in \{1,2,3, n-1, n\}\}$ ✓

$P_2 = \{CJ_i; i \in \{1, 3, n-2, n\}\}$ ✗

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \dots, CJ_{n-2} CJ_{n-1}, CJ_n$$

$I = \{CJ_i; i \in \{2,3, n-1\}\}$ ⟶ $O$

$P_0 = \{CJ_i; i \in \{0,1,3, n-2\}\}$ ✗

$P_1 = \{CJ_i; i \in \{1,2,3, n-1, n\}\}$ ✓

$P_2 = \{CJ_i; i \in \{1, 3, n-2, n\}\}$ ✗

# Fyookball and Lundeberg (2019)

## CashFusion

Authors: Jonald Fyookball, Mark B. Lundeberg

### Introduction

**THE PROBLEM:**

CashShuffle is a powerful tool for obfuscating the origin of a coin. However, after shuffling a wallet, a user will inevitably wish to consolidate several coins, and for this another tool is needed. We need a method to coordinate coinjoin transactions with multiple inputs per user. This is inherently challenging because we want to hide input linkages while simultaneously attempting to blame/ban users who don't sign all their inputs.

https://github.com/cashshuffle/spec/blob/master/CASHFUSION.md

# Questions?

# The problem(s)

$$CJ_0, CJ_1\ CJ_2, CJ_3, \ldots, CJ_{n-2}CJ_{n-1}, CJ_n$$

# The problem(s)

$$W_0, W_1\ W_2, W_3, \ldots, W_{n-2} W_{n-1}, W_n$$