grincon0
18.11.09 // c-base berlin

Dandelion in Grin: Privacy-Preserving Transaction Aggregation & Propagation

@quentinlesceller

Outline

Grin
- Transactions
- Blockchain
- Open Problems

Dandelion
- Current problems in P2P network
- What is Dandelion
- Key terminology

Grin & Dandelion
- Dandelion Implementation in Grin
- Solving the problem
- Transaction Aggregation

Grin

What is MW/Grin?

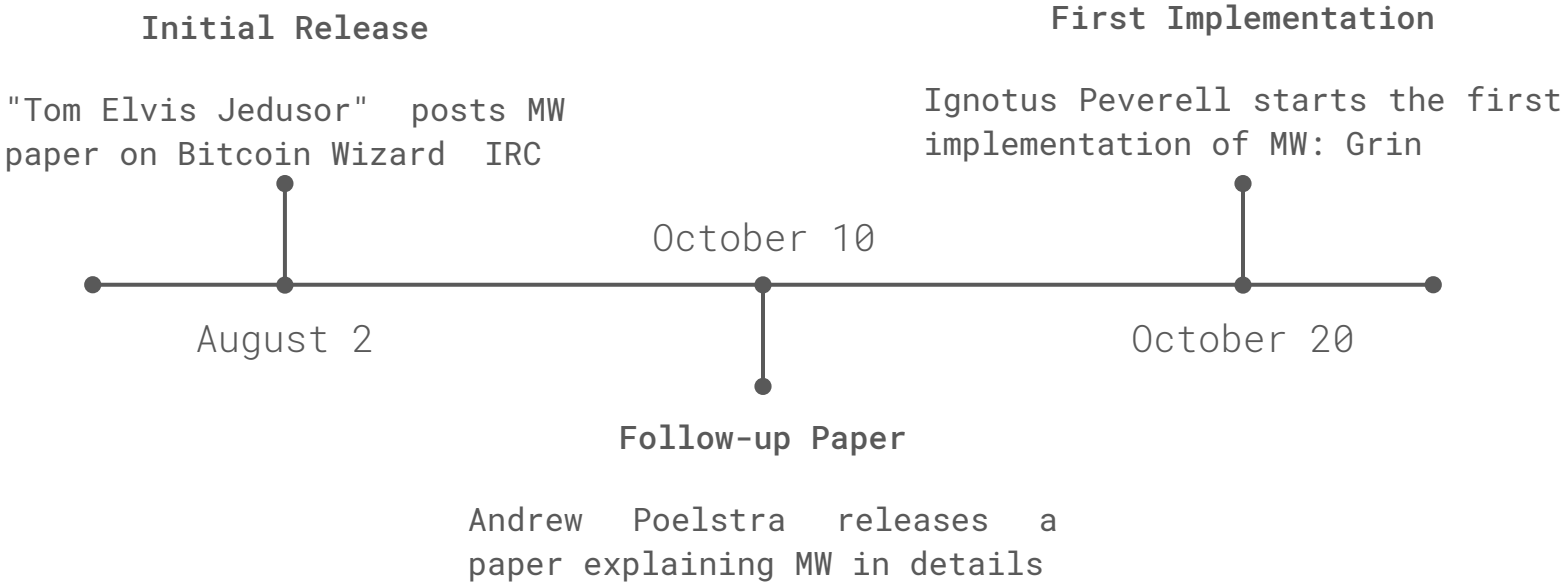
- MimbleWimble (MW) is a completely new blockchain design that offers several benefits:
  - Privacy by default
  - Massively prunable
  - Relies on proven elliptic curve cryptography

- Output-based like Bitcoin but without script

- Grin is the first implementation of the protocol

History

2016



Grin Transactions

- In Bitcoin, every output has a script (script pubkey) attached to it. In order to spend one of them, conditions in the script must be met.

- In MW/Grin outputs only have public keys: no script.

- Hence MW/Grin transactions are scriptless.

Grin Transactions

A Grin transaction contains the following parts:

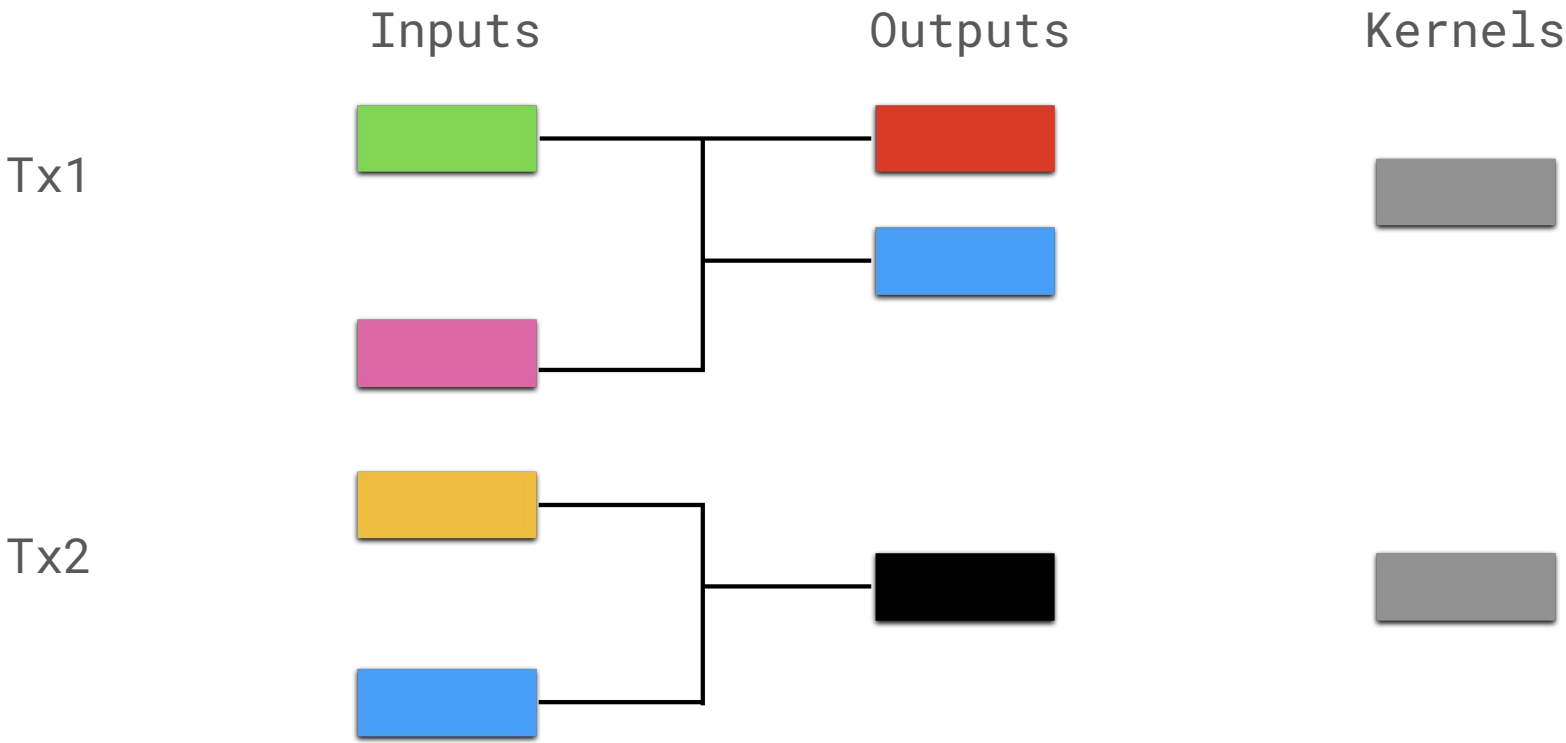
- Inputs (reference to old outputs)
- Outputs: confidential transactions (amounts are blinded) + range proofs
- Kernel: difference between outputs and inputs + fee + signature

Grin Transactions

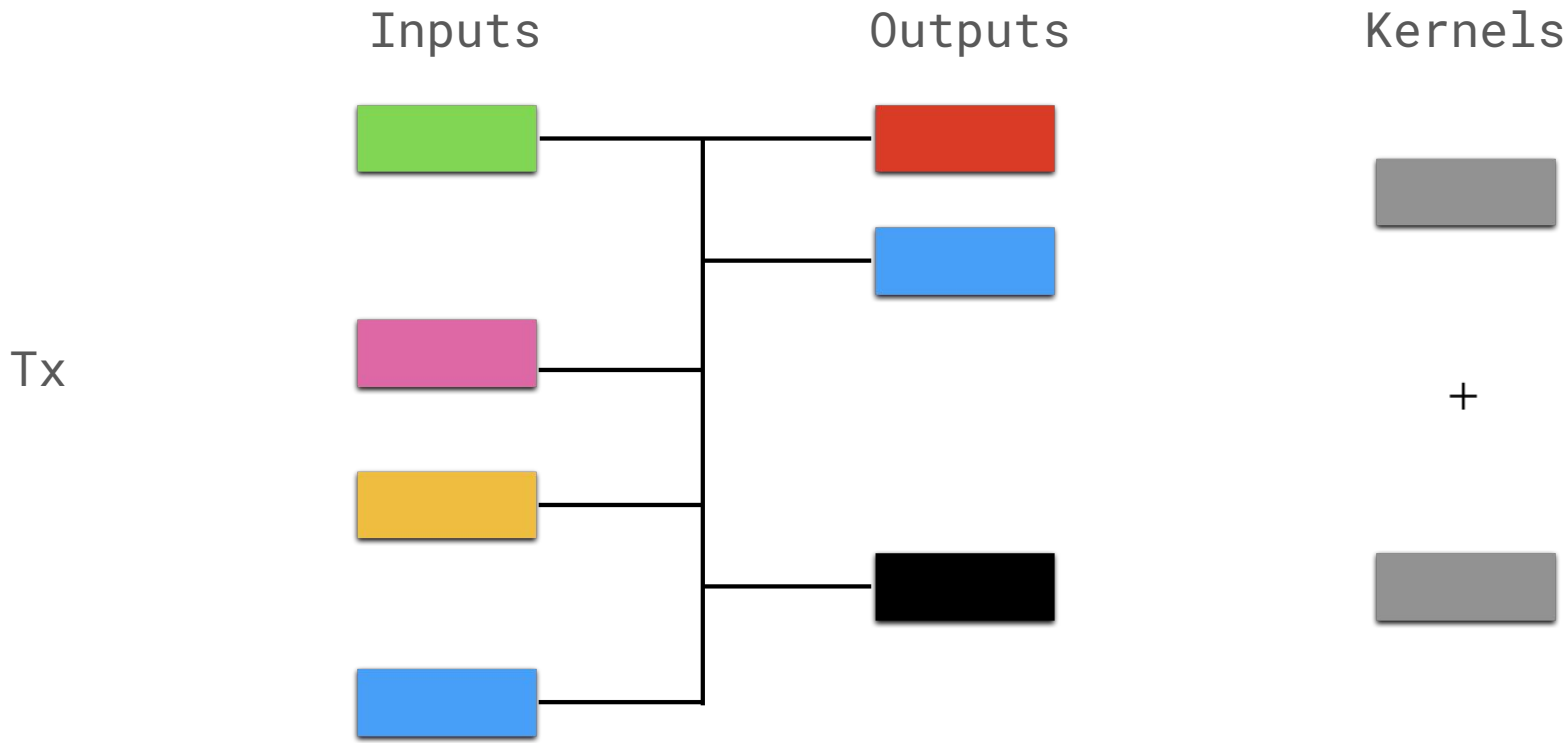
A Grin transaction validation is based on two properties:

- Verification of zero sums.
- Possession of private keys.

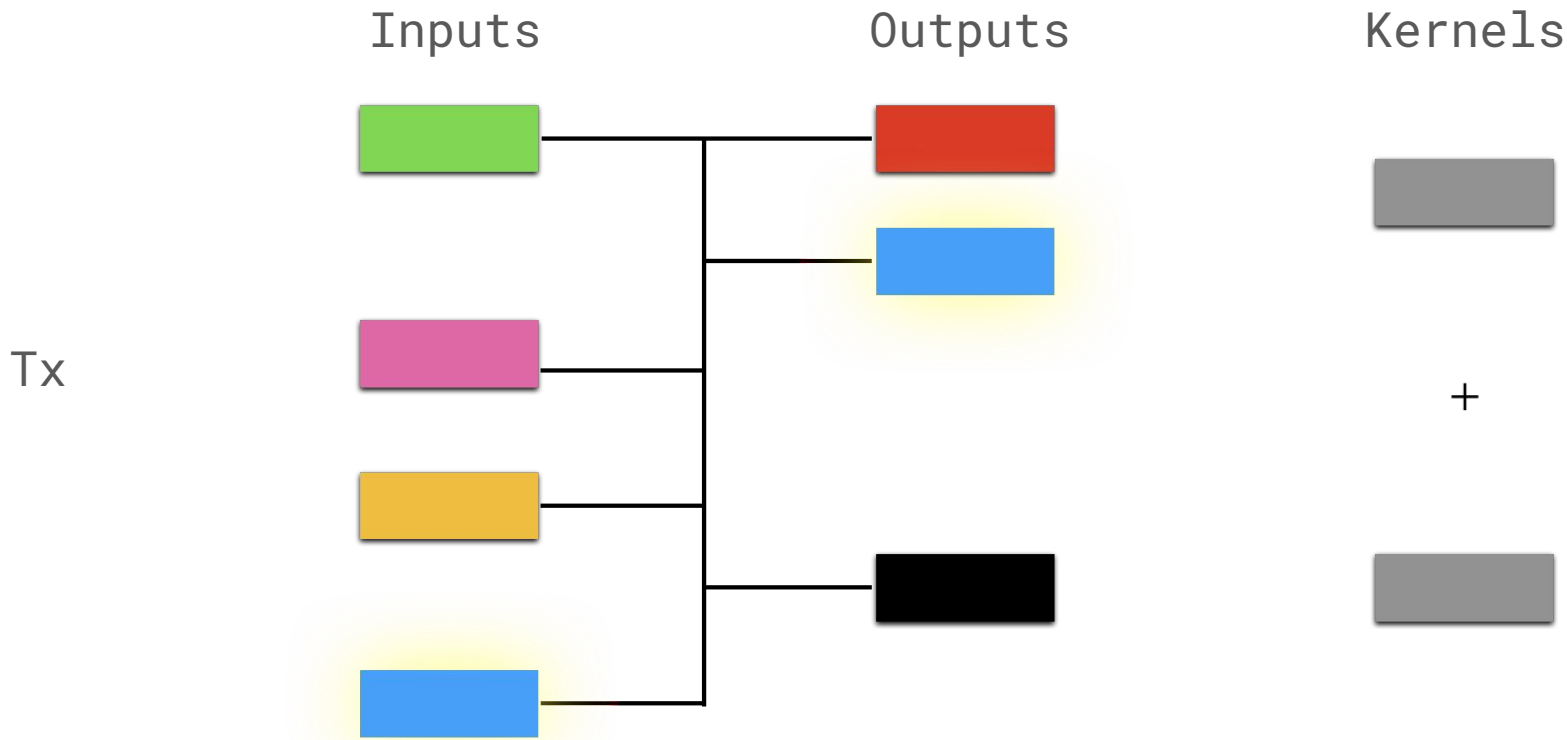
Grin Transactions



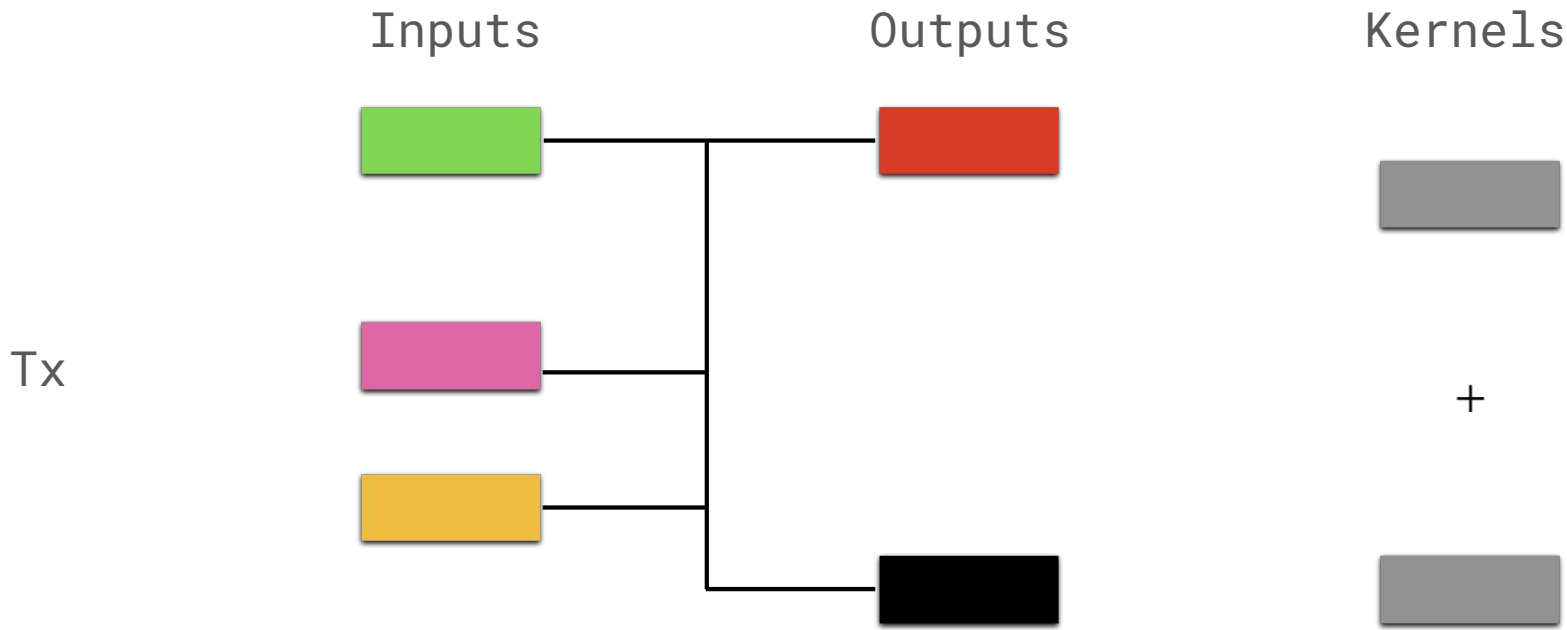
Grin Transactions



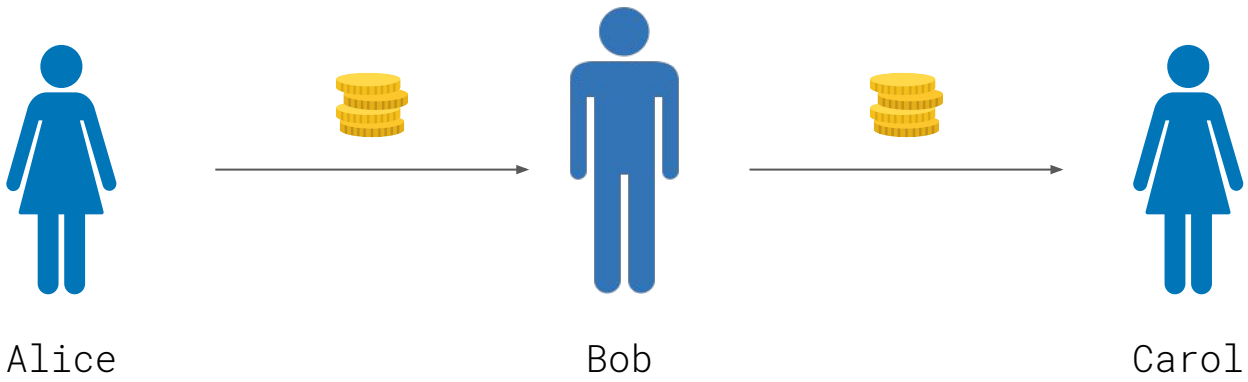
Grin Transactions



Grin Transactions

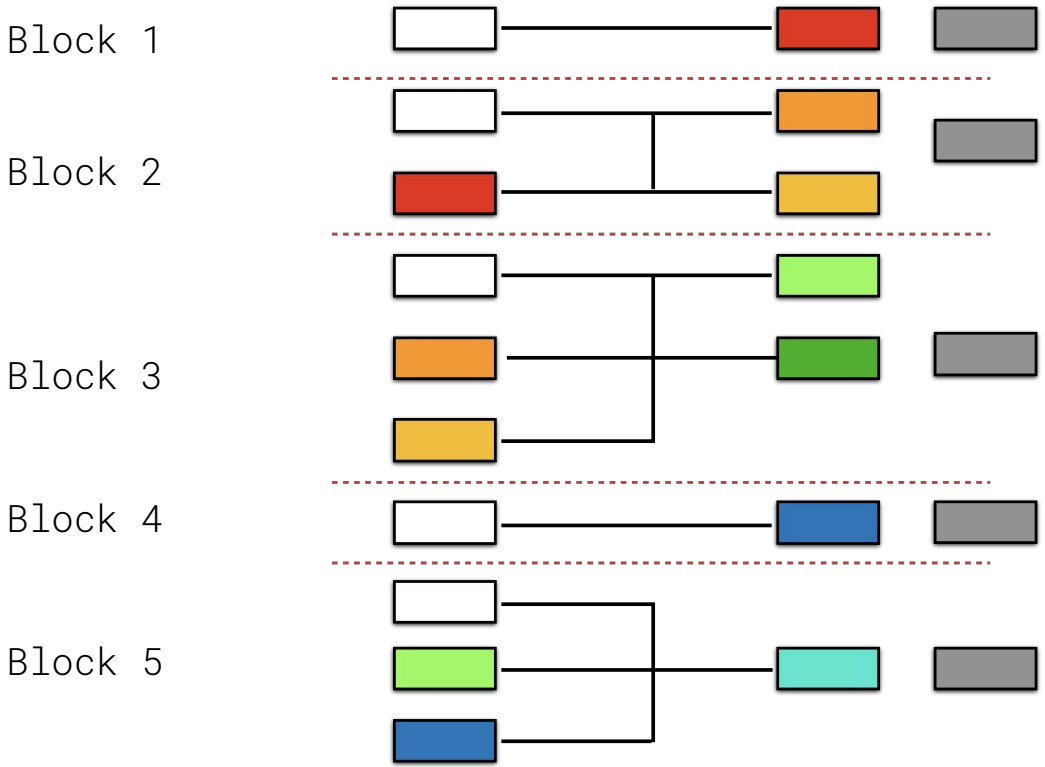


Grin Transactions



Grin Transactions

Grin Blockchain



Grin Blockchain

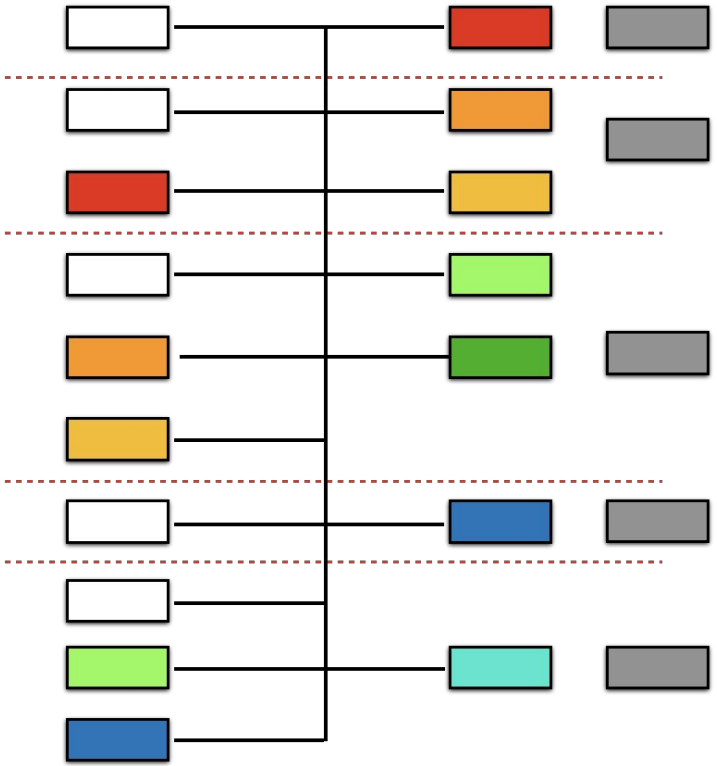
Block 1

Block 2

Block 3

Block 4

Block 5



Grin Blockchain

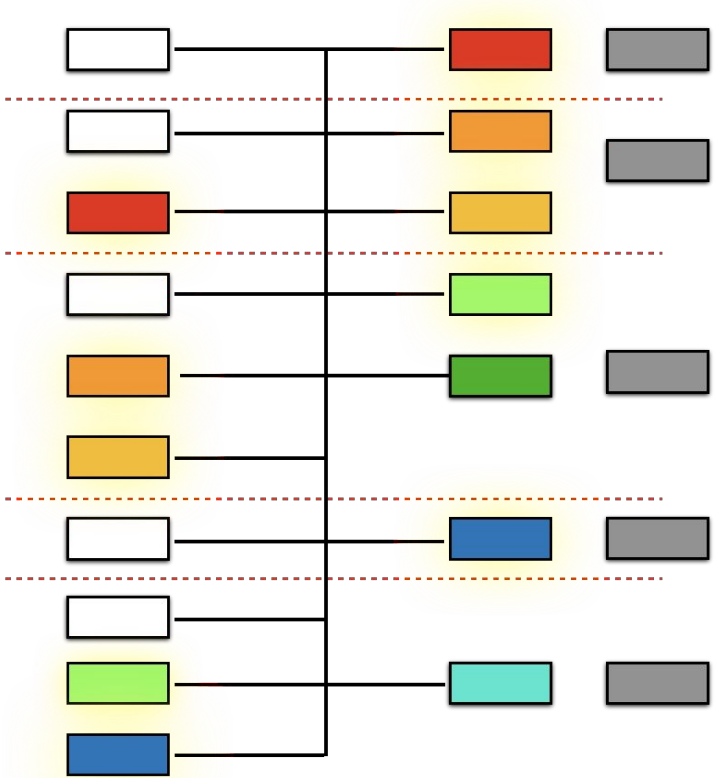
Block 1

Block 2

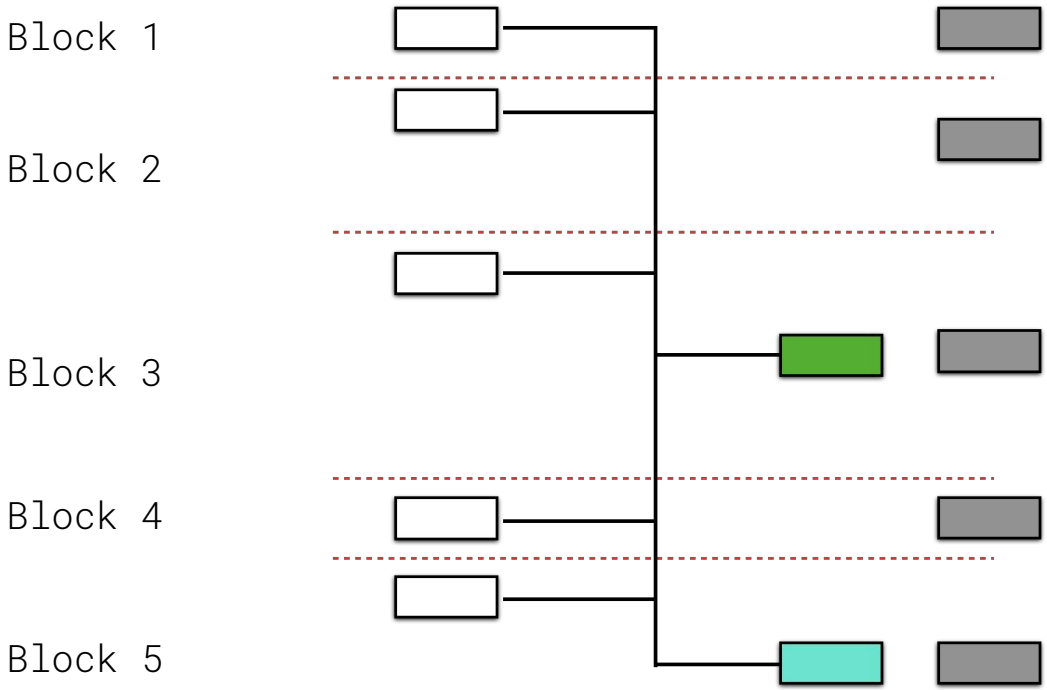
Block 3

Block 4

Block 5

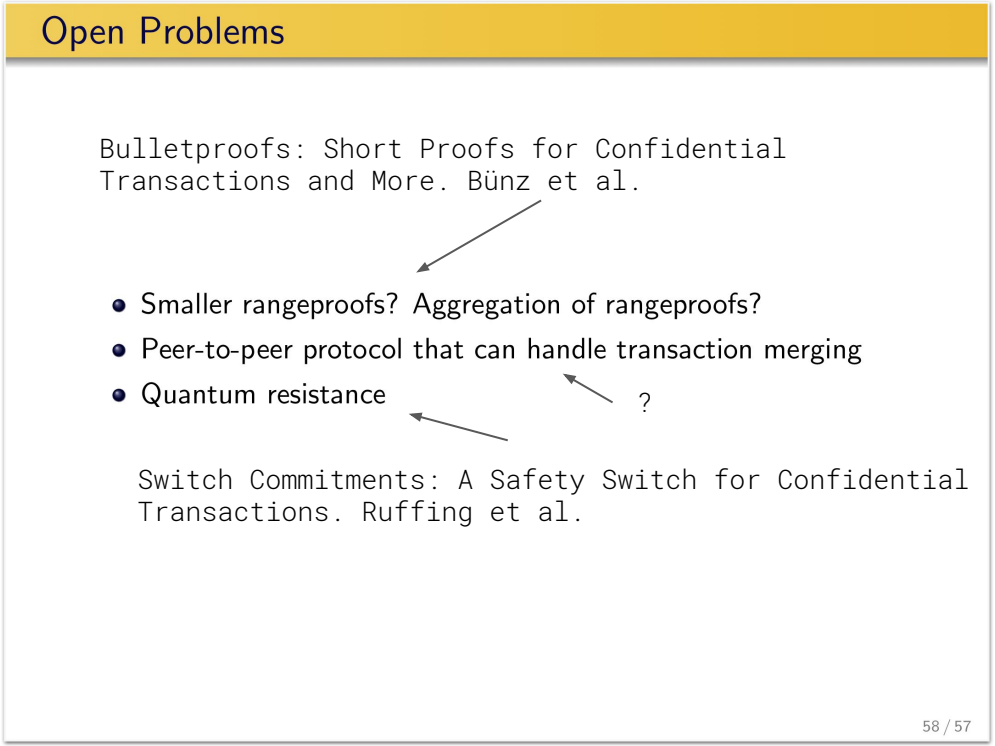


Grin Blockchain



Open Problems

- Mentioned by Andrew Poelstra at Scaling Bitcoin 2016



Dandelion

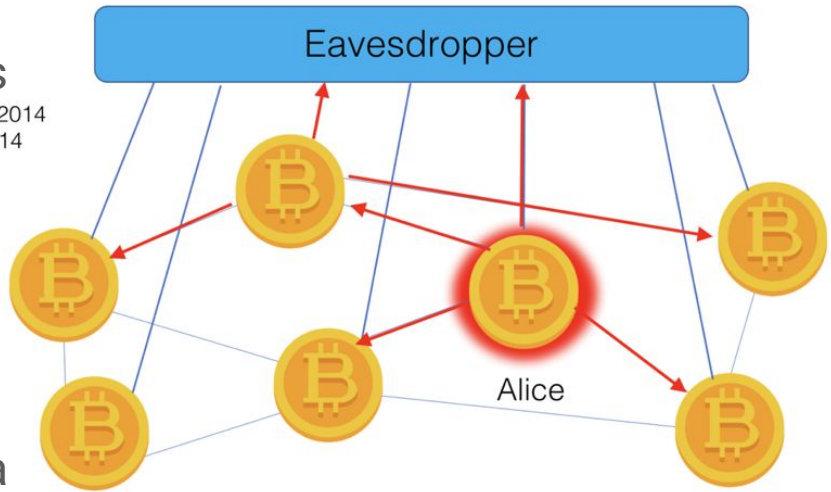
Problem in the current P2P network

- The P2P network currently forwards content in a structured way that allows observers to deanonymize users

Biryukov et al., 2014
Koshy et al., 2014

- An eavesdropper is capable to assign an IP to a bitcoin address

- Identify the first sender of a Bitcoin transaction

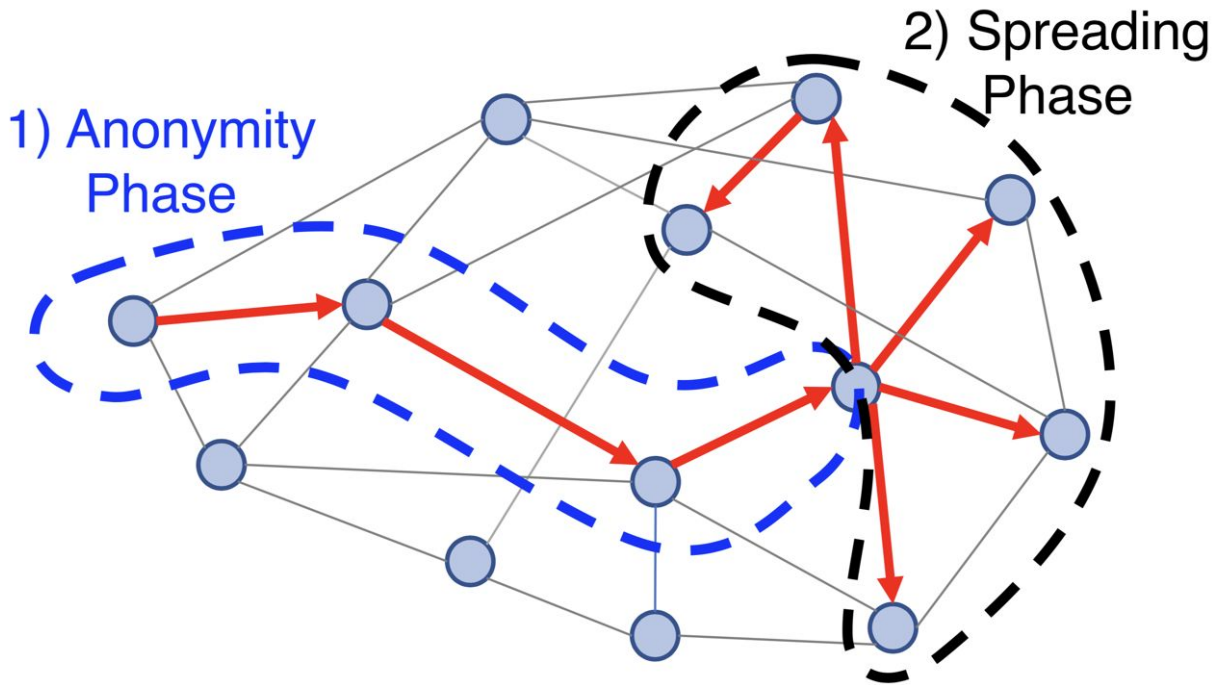


What is Dandelion?

- Dandelion is a simple networking policy to improve the anonymity in a peer-to-peer network.
- Created by Shaileshh Bojja, Giulia Fanti and Pramod Viswanath (University of Illinois).
- A BIP was written by Giulia Fanti, Andrew Miller, Surya Bakshi, Shaileshh Bojja and Pramod Viswanat.

https://github.com/dandelion-org/bips/blob/master/bip-dandelion.mediawiki

What is Dandelion?

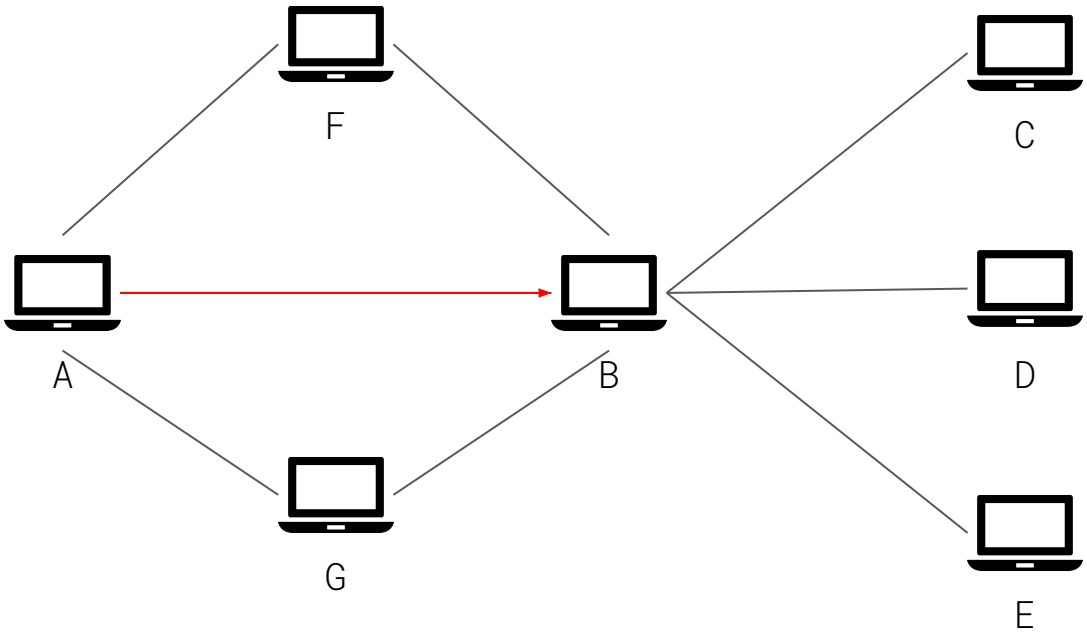


Key Terminology

Stem transaction	A transaction during the anonymity phase.
Stem memory pool	Also called stempool. The pool where the stem transactions are stored.
Dandelion relay	The peer chosen to relay the stem transaction. Updated every X minutes.
To stem	To broadcast to the Dandelion relay only.
To fluff	To broadcast to all the peers normally

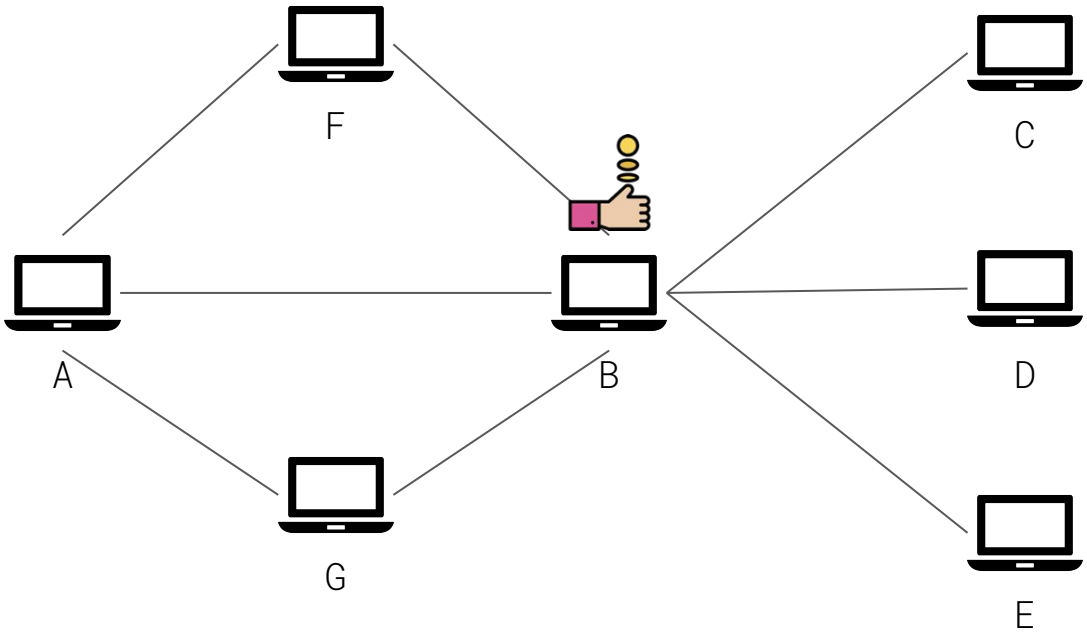
Dandelion Step-by-Step

Step 1: A sends a Stem Transaction to B



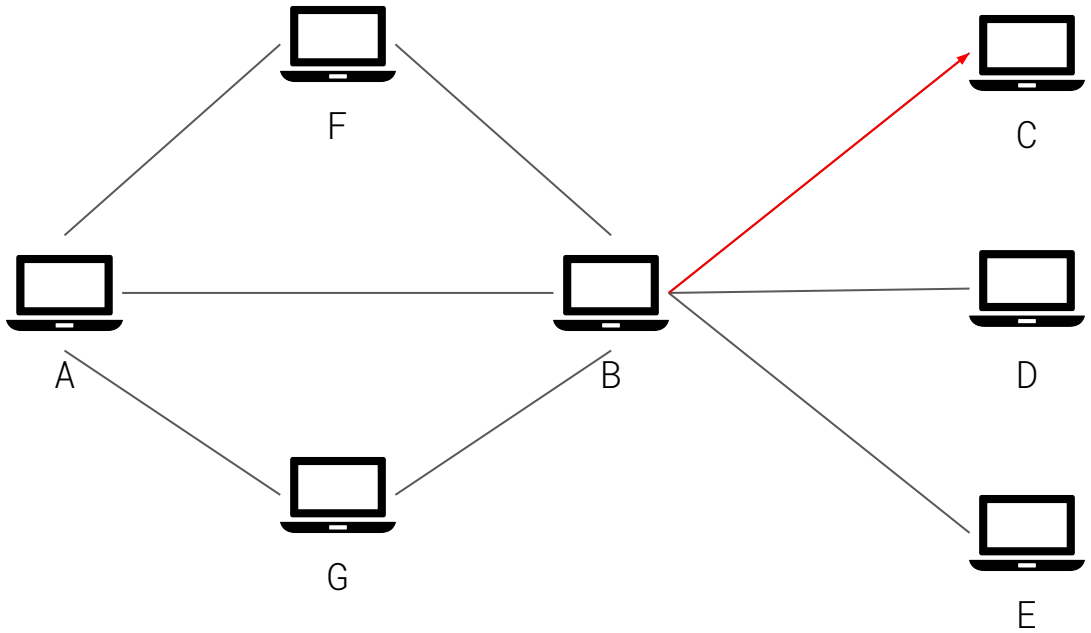
Dandelion Step-by-Step

Step 2: B receives the Stem Tx and do a coin flip



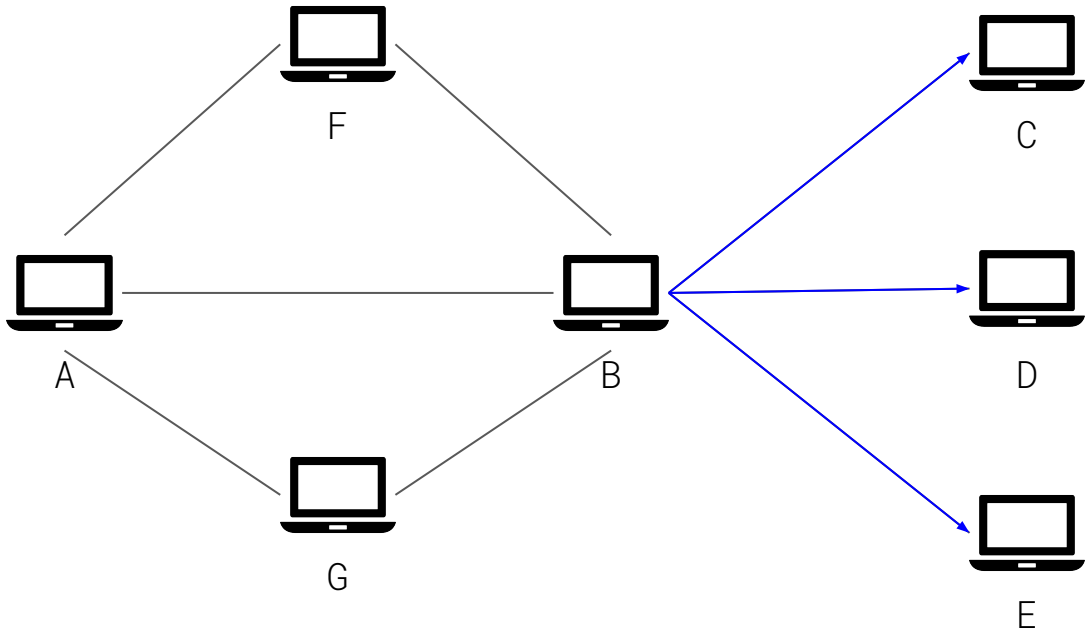
Dandelion Step-by-Step

Step 3: B sends the stem transaction to its Dandelion Relay



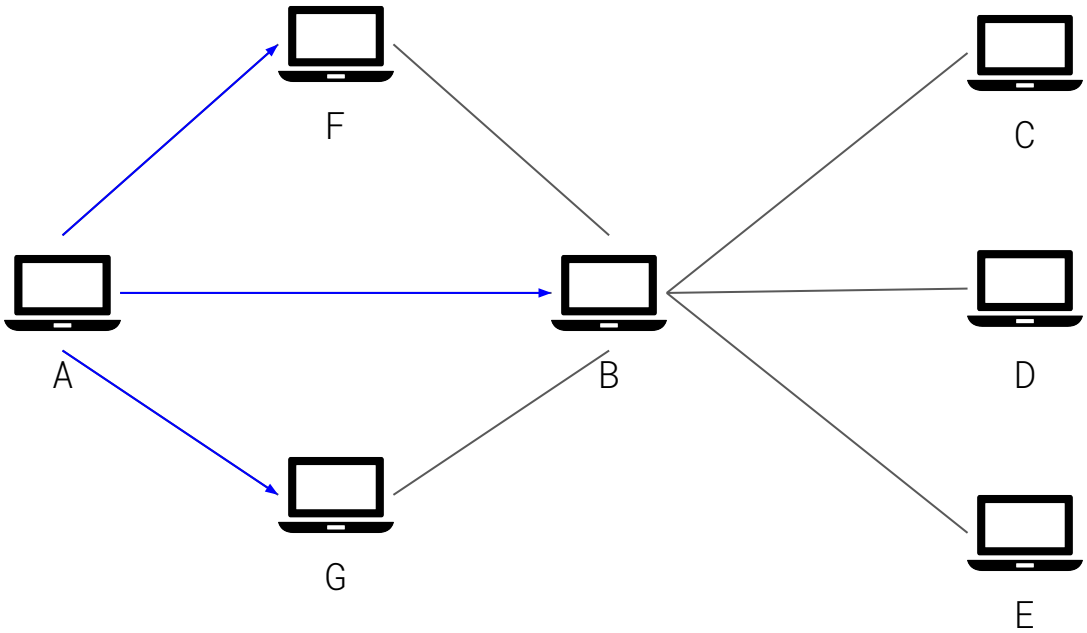
Dandelion Step-by-Step

Step 3: B sends the transaction to all its peers



Dandelion Step-by-Step

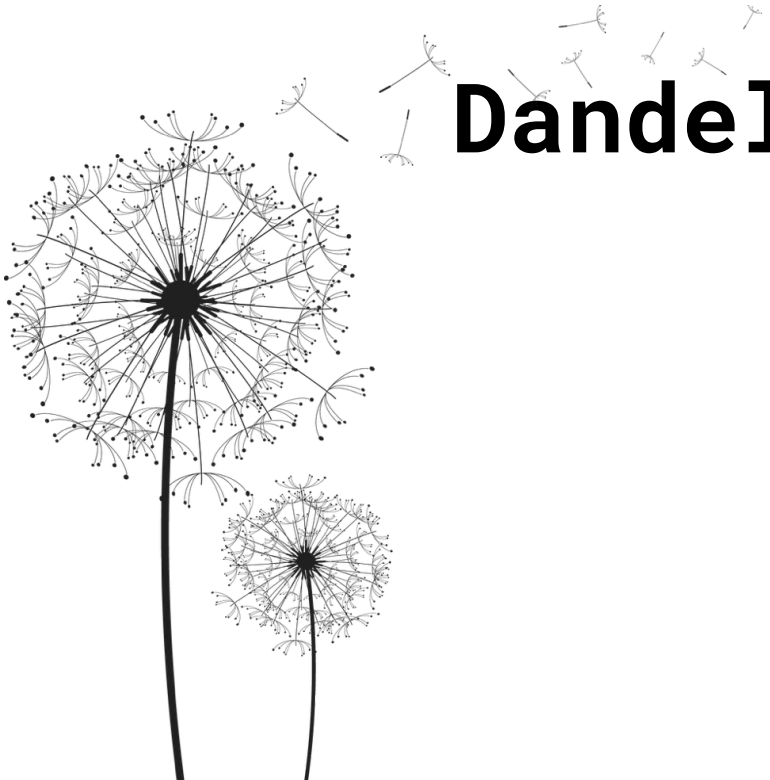
Step X: B does not play by the rules.



What is Dandelion?

3 parameters:

- The Dandelion probability (by default 90%)
- The time between re-randomizations of the Dandelion relay (every 10 minutes)
- The embargo timer (3 minutes)

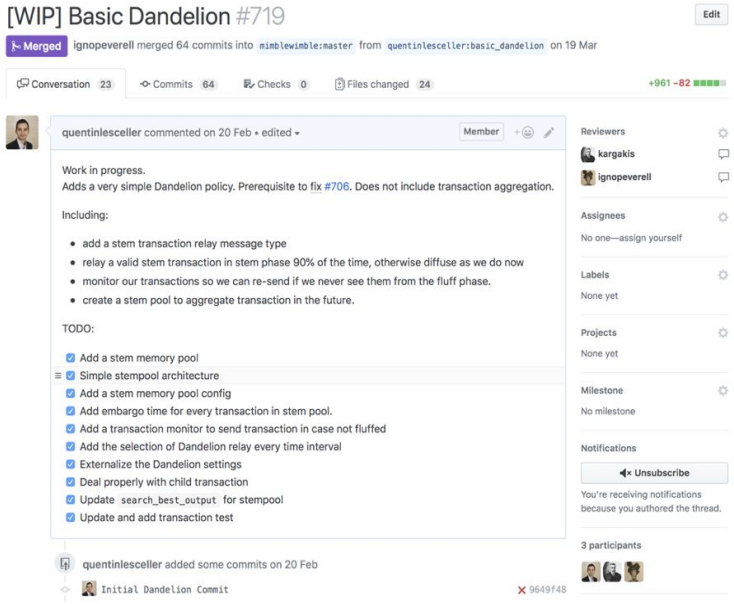


Dandelion in Grin

Dandelion Implementation in Grin

Basic implementation first

- Implemented in Grin in February.
- Work in collaboration with Giulia Fanti's team.
- In the testnet2 release.



Solving the problem

How can we aggregate transactions trustlessly in a peer-to-peer protocol?

- No consensus on the content of the memory pool.
- Nodes should not be able to find out the transaction graph.

Can we use the Dandelion protocol to aggregate tx in Grin?

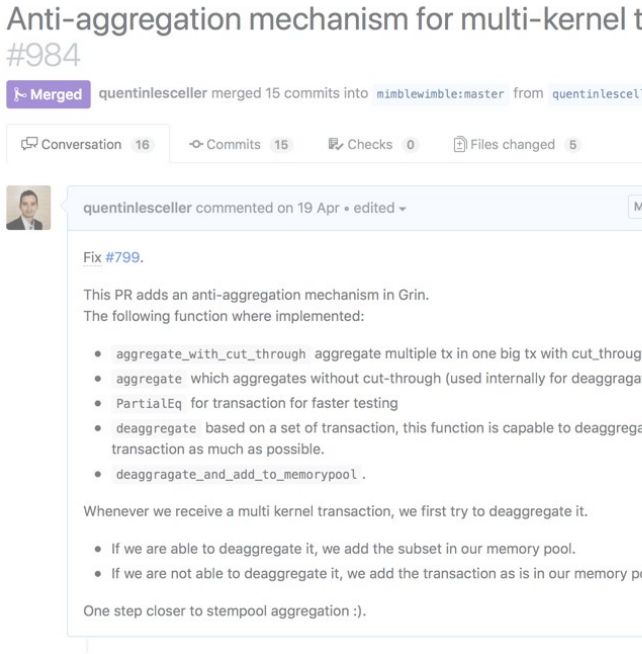
Solving the problem

- We can aggregate transactions during the stem phase.
- Stop aggregating when fluff.
- New concept of patience: How long the node will wait for other transactions to aggregate before broadcasting again.

Implementing Transaction Aggregation in Grin

- Could cause a denial of service. (Aggregation with low fee transactions then rejected by miners)

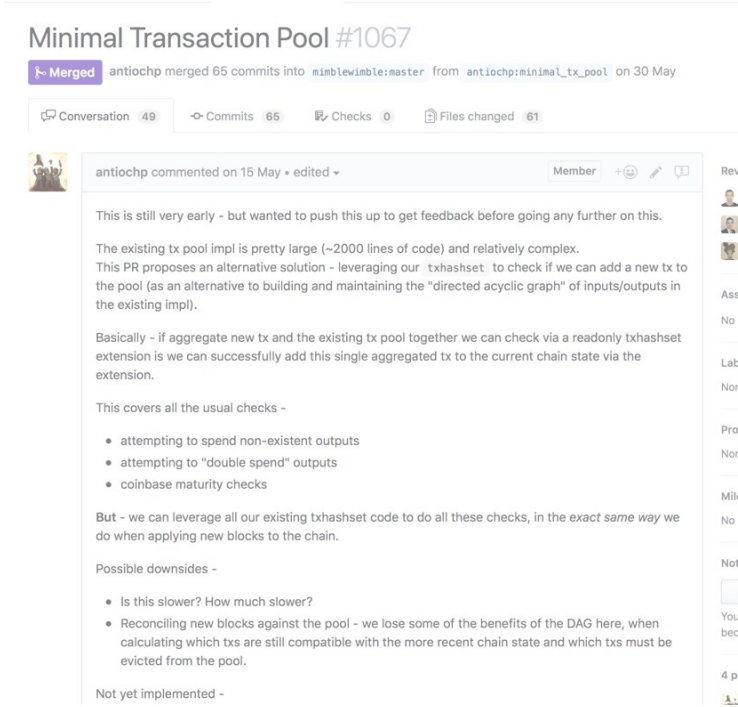
- Need to deploy some countermeasure (antiaggregation mechanism) PR #985
  - Basically, when a node receive an aggregated transaction, try to deaggregate it using other tx in its mempool.



Implementing Transaction Aggregation in Grin

Implemented in Grin end of May.
PR #1067

Integrated since testnet2 release.



Minimal Transaction Pool #1067

Merged antiochp merged 65 commits into mimblewimble:master from antiochp:minimal_tx_pool on 30 May

Conversation 49 Commits 65 Checks 0 Files changed 61

antiochp commented on 15 May • edited Member

This is still very early - but wanted to push this up to get feedback before going any further on this.

The existing tx pool impl is pretty large (~2000 lines of code) and relatively complex.
This PR proposes an alternative solution - leveraging our `txhashset` to check if we can add a new tx to the pool (as an alternative to building and maintaining the "directed acyclic graph" of inputs/outputs in the existing impl).

Basically - if aggregate new tx and the existing tx pool together we can check via a readonly txhashset extension is we can successfully add this single aggregated tx to the current chain state via the extension.

This covers all the usual checks -

- attempting to spend non-existent outputs
- attempting to "double spend" outputs
- coinbase maturity checks

But - we can leverage all our existing txhashset code to do all these checks, in the *exact same way* we do when applying new blocks to the chain.

Possible downsides -

- Is this slower? How much slower?
- Reconciling new blocks against the pool - we lose some of the benefits of the DAG here, when calculating which txs are still compatible with the more recent chain state and which txs must be evicted from the pool.

Not yet implemented -

Implementing Transaction Aggregation in Grin

Node sends a transaction on the network, it's directly broadcasted to the next Dandelion relay as stem transaction.

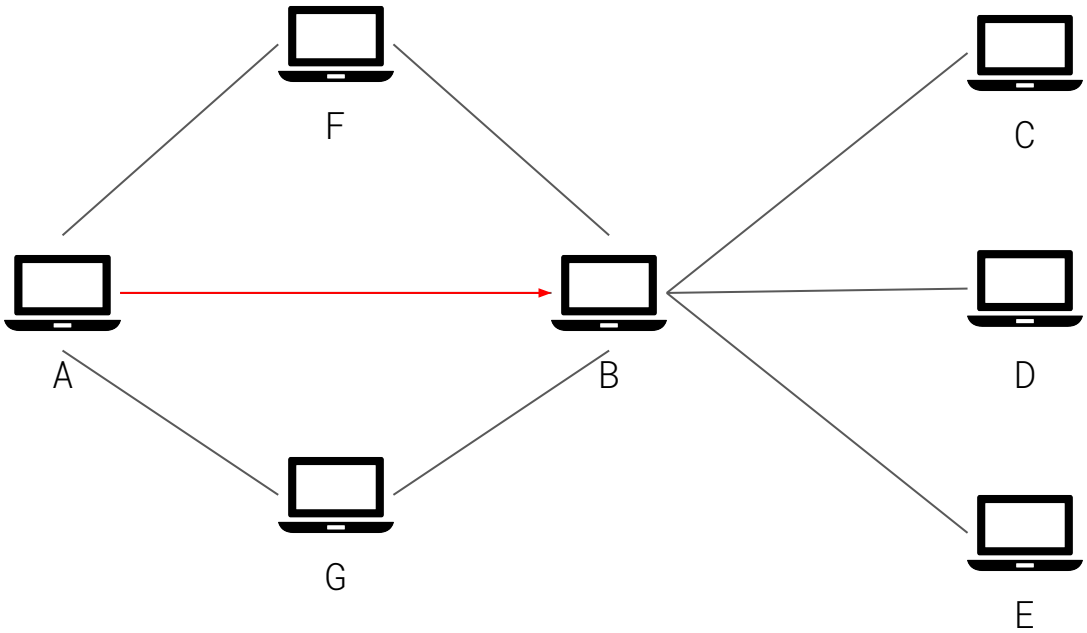
The Dandelion relay waits a period of time (the patience timer), in order to get more stem transactions to aggregate.

At the end of the timer, the relay does a coin flip for each new stem transaction and determines if it will stem it (send to the next Dandelion relay) or fluff it (broadcast normally).

Then the relay will take all the transactions to stem, aggregate them, and broadcast them to the next Dandelion relay. Same for the transactions to fluff, except that it will broadcast the aggregated transactions “normally” (to a random subset of the peers).

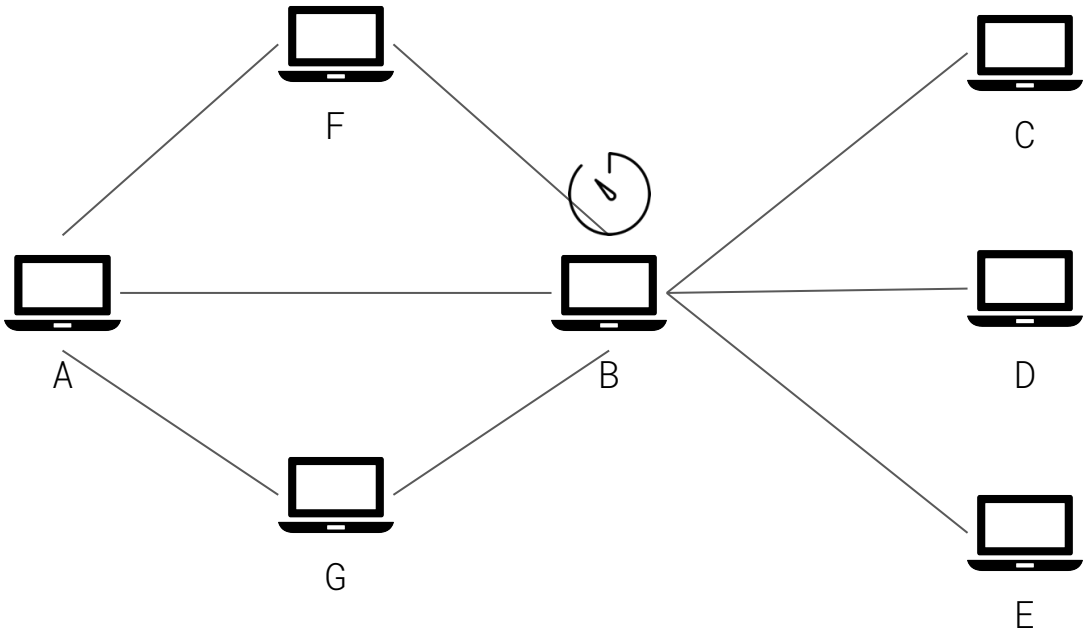
Grin Dandelion: Step-by-Step

Step 1: A sends a Stem Transaction to B



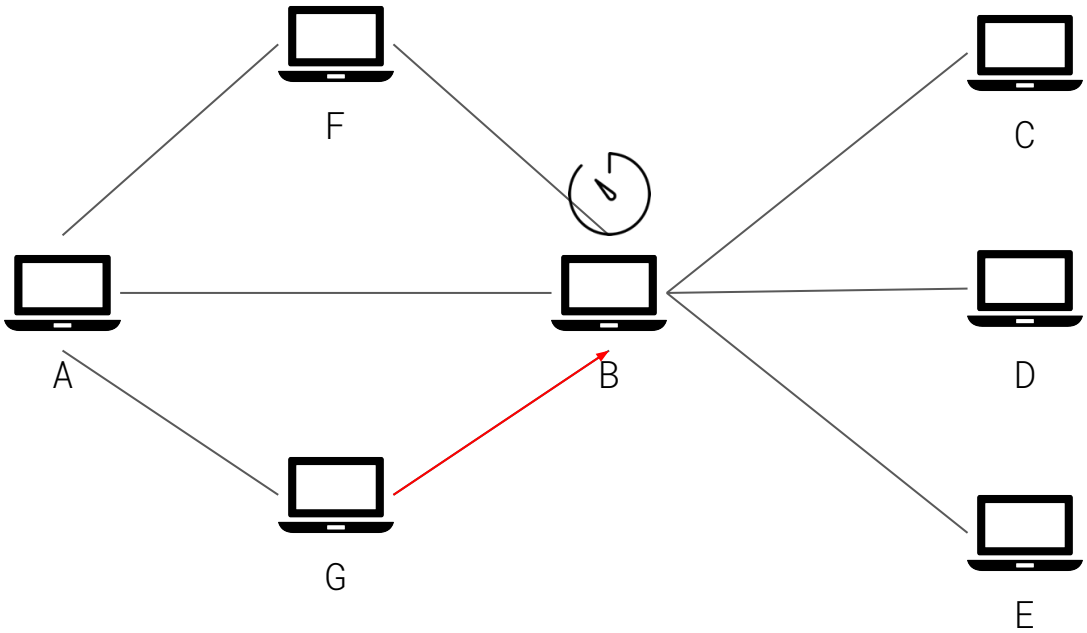
Grin Dandelion: Step-by-Step

Step 2: B waits



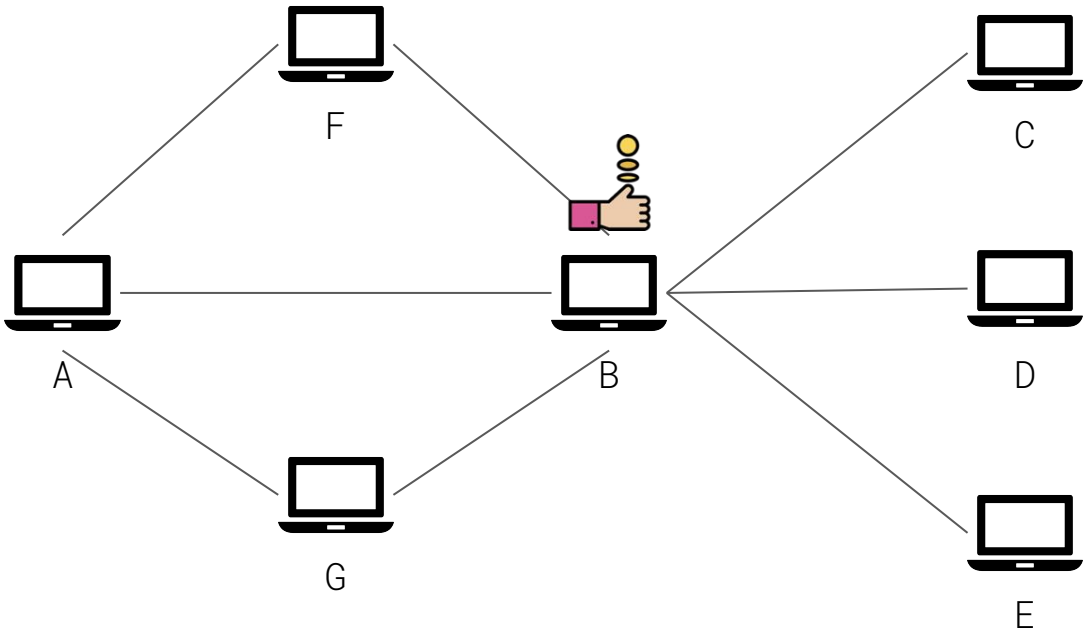
Grin Dandelion: Step-by-Step

Step 2: B waits



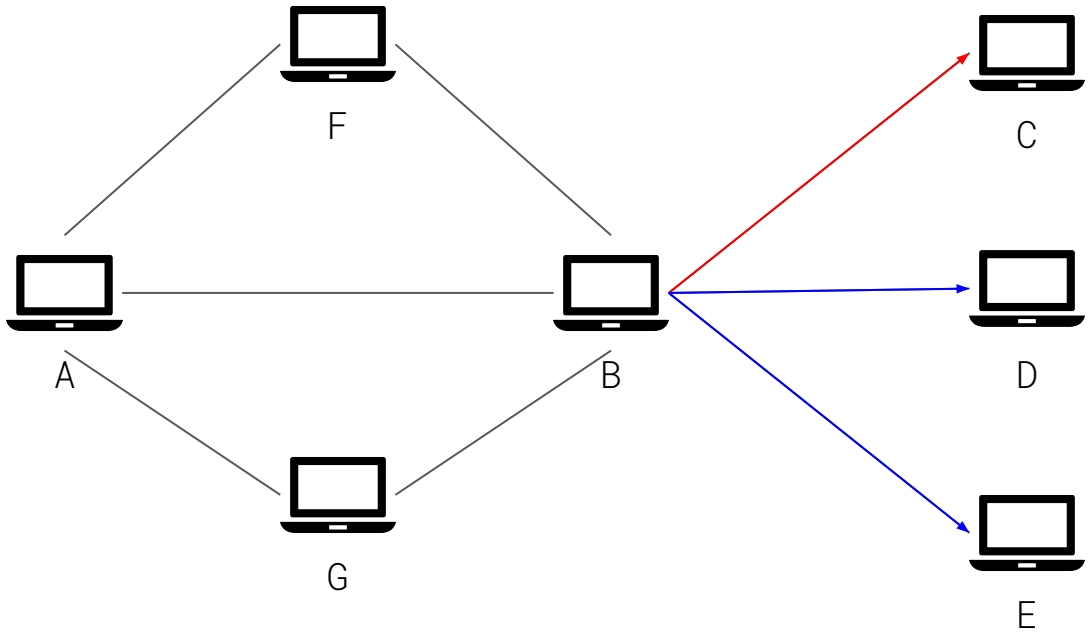
Grin Dandelion: Step-by-Step

Step 3: B does a coin flip for each new stem transaction and marks them as “To Stem” or “To Fluff”



Grin Dandelion: Step-by-Step

Step 4: B broadcast aggregated transactions



Aggregated "To Stem" tx

Aggregated "To Fluff" tx

Current challenges and future work

- Hard to simulate a whole network
- Quantify how many “natural” aggregations happens on the network
- Test our parameters

References

- https://github.com/mimblewimble/grin - Code Repository
- https://www.grin-forum.org - Forum and Links to Resources
- https://gitter.im/grin_community/ - Public and Dev Chat
- https://github.com/gfanti/bips/blob/master/bip-dandelion.mediawiki - Dandelion BIP
- https://github.com/mimblewimble/grin/pull/719 - Initial Implementation of Dandelion protocol in Grin
- https://github.com/mimblewimble/grin/pull/1023 - Work-in-progress implementation of the aggregation in Grin
- https://github.com/mimblewimble/grin/pull/1067 - Minimal Transaction Pool
- A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in bitcoin p2p network. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pages 15–29. ACM, 2014.
- G. C. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath. Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2(2):29:1–29:35, June 2018.
- Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Efficient range proofs for confidential transactions. Cryptology ePrint Archive, Report 2017/1066, 2017. https://eprint.iacr.org/2017/1066.
- Ruffing, T., & Malavolta, G. (2017, April). Switch Commitments: A Safety Switch for Confidential Transactions. In International Conference on Financial Cryptography and Data Security (pp. 170-181). Springer, Cham.

https://grin-tech.org