Mimblewimble: Private, Massively-Prunable Blockchains

Andrew Poelstra

grindelwald@wpsoftware.net

October 8, 2016

History

- 04:30 UTC, August 2nd, 2016: “Tom Elvis Jedusor” posts a .onion link to a text file on IRC, titled MIMBLEWIMBLE and dated July 19.

History

- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC, titled MIMBLEWIMBLE and dated July 19.
- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.

History

- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC, titled MIMBLEWIMBLE and dated July 19.
- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.
- (Previous week: I had independently found a small part of Mimblewimble which was almost supported by Elements Alpha; starting from this I got the gist of the paper.)

History

- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC, titled MIMBLEWIMBLE and dated July 19.
- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.
- (Previous week: I had independently found a small part of Mimblewimble which was almost supported by Elements Alpha; starting from this I got the gist of the paper.)
- Following week: discussion on Reddit with Greg Sanders and others leads to understanding Mimblewimble's trust model, and hints that the new crypto has merit.

History

- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC, titled MIMBLEWIMBLE and dated July 19.
- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.
- (Previous week: I had independently found a small part of Mimblewimble which was almost supported by Elements Alpha; starting from this I got the gist of the paper.)
- Following week: discussion on Reddit with Greg Sanders and others leads to understanding Mimblewimble's trust model, and hints that the new crypto has merit.
- September: myself and Avi Kulkarni develop an extension, "sinking signatures", to greatly improve its scaling properties.

History

- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC, titled MIMBLEWIMBLE and dated July 19.
- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.
- (Previous week: I had independently found a small part of Mimblewimble which was almost supported by Elements Alpha; starting from this I got the gist of the paper.)
- Following week: discussion on Reddit with Greg Sanders and others leads to understanding Mimblewimble's trust model, and hints that the new crypto has merit.
- September: myself and Avi Kulkarni develop an extension, "sinking signatures", to greatly improve its scaling properties.
- I am not Lord Voldemort.

What is Mimblewimble?

- Mimblewimble is a design for a blockchain-based ledger that is very different from Bitcoin.

What is Mimblewimble?

- Mimblewimble is a design for a blockchain-based ledger that is very different from Bitcoin.
- It can be implemented as a sidechain, or softforked into Bitcoin (with limitations).

What is Mimblewimble?

- Mimblewimble is a design for a blockchain-based ledger that is very different from Bitcoin.
- It can be implemented as a sidechain, or softforked into Bitcoin (with limitations).
- In Bitcoin transactions, old outputs sign new outputs; outputs have "script pubkeys" that are independent of each other. In Mimblewimble transactions, outputs have only EC pubkeys, and the difference between new outputs' keys and old ones' is multisigned by all transacting parties.

What is Mimblewimble?

- Mimblewimble is a design for a blockchain-based ledger that is very different from Bitcoin.
- It can be implemented as a sidechain, or softforked into Bitcoin (with limitations).
- In Bitcoin transactions, old outputs sign new outputs; outputs have "script pubkeys" that are independent of each other. In Mimblewimble transactions, outputs have only EC pubkeys, and the difference between new outputs' keys and old ones' is multisigned by all transacting parties.
- Mimblewimble transactions are inherently scriptless.

Mimblewimble Transactions

A Mimblewimble transaction is the following data:

- Inputs (references to old outputs).

Mimblewimble Transactions

A Mimblewimble transaction is the following data:

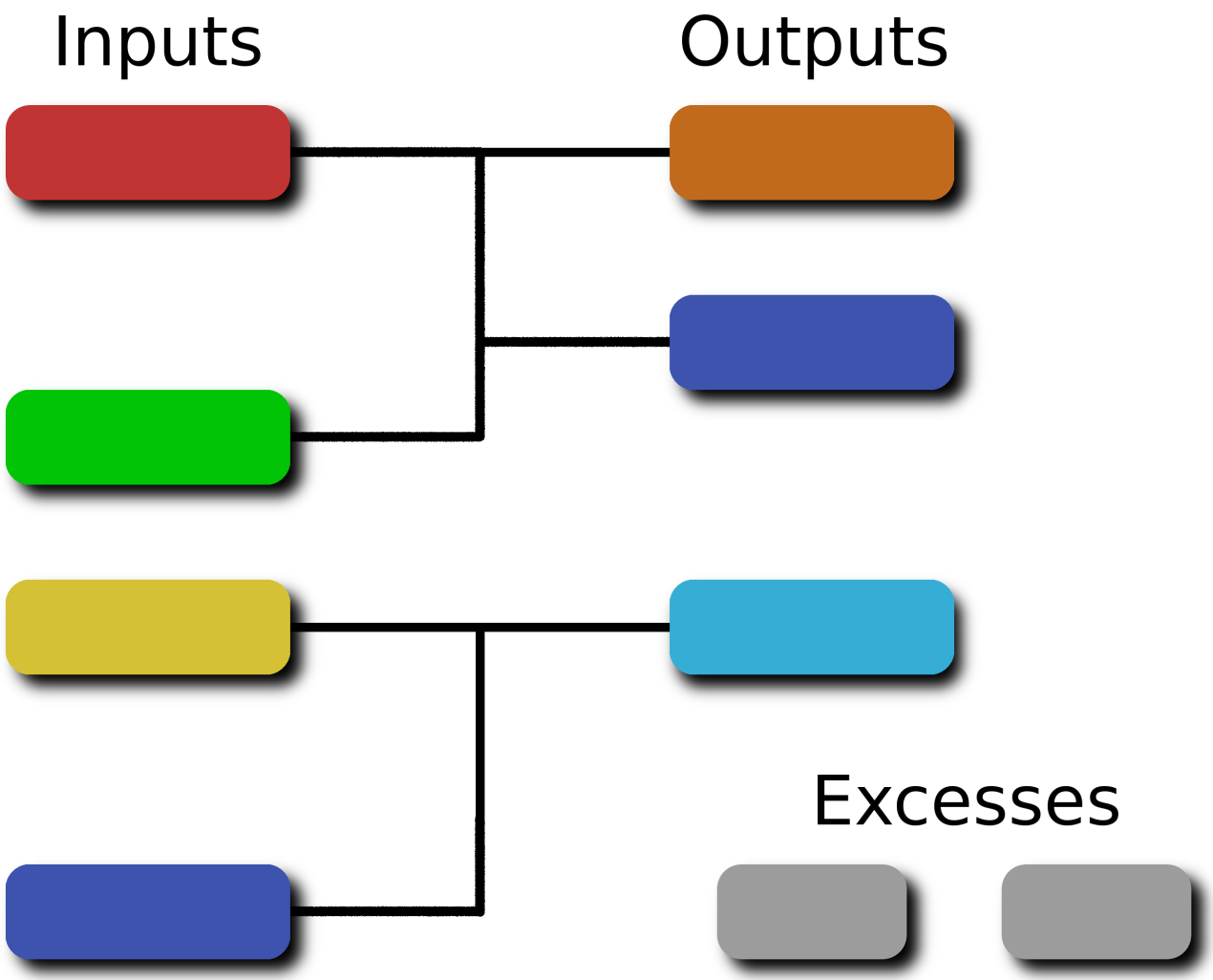
- Inputs (references to old outputs).
- Outputs: confidential transaction outputs (group elements, which blind and commit to amounts), plus rangeproofs.

Mimblewimble Transactions

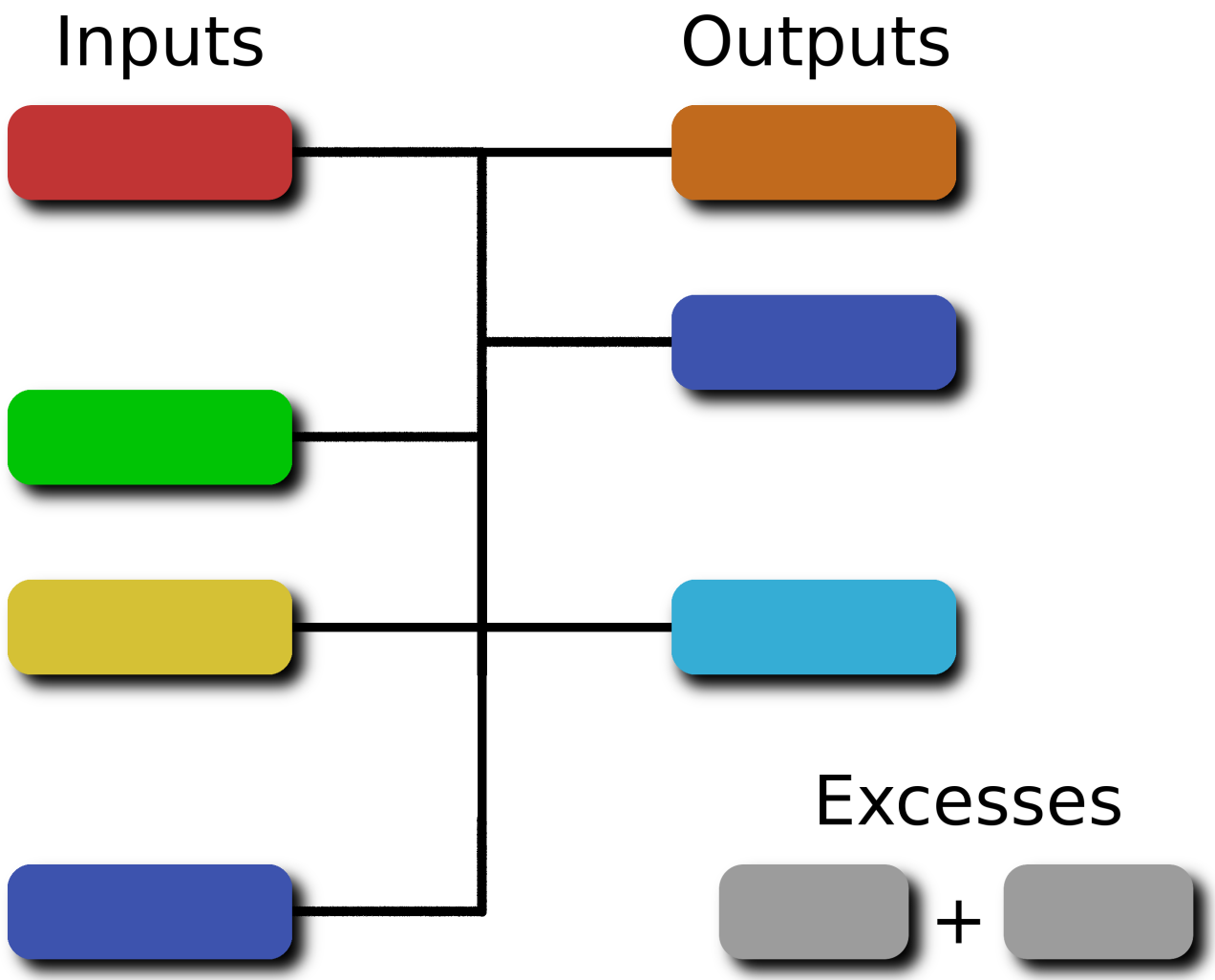
A Mimblewimble transaction is the following data:

- Inputs (references to old outputs).
- Outputs: confidential transaction outputs (group elements, which blind and commit to amounts), plus rangeproofs.
- Excess: difference between outputs and inputs (group element), plus signature (for authentication and to prove non-infaltion)

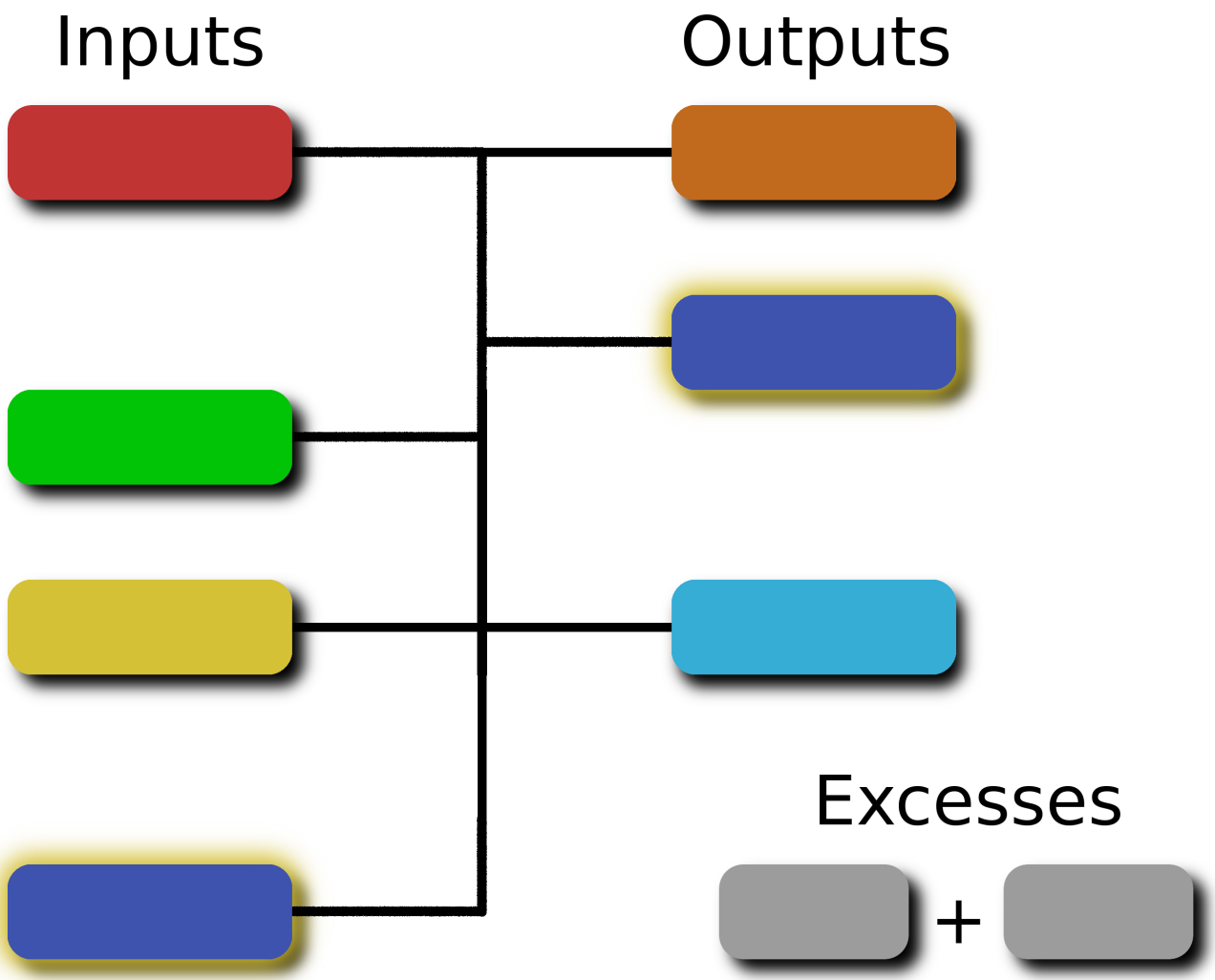
Mimblewimble Transactions



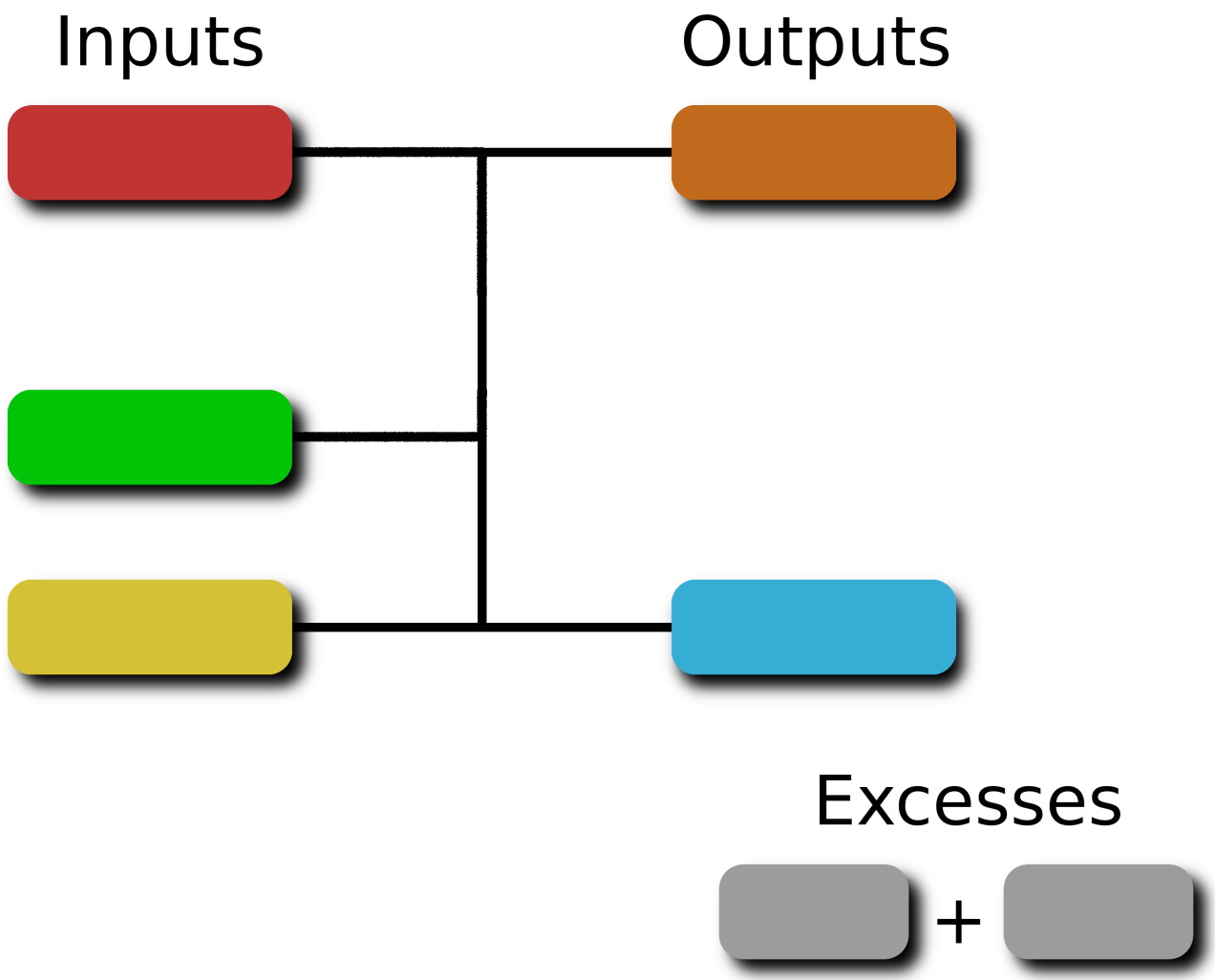
Mimblewimble Transactions



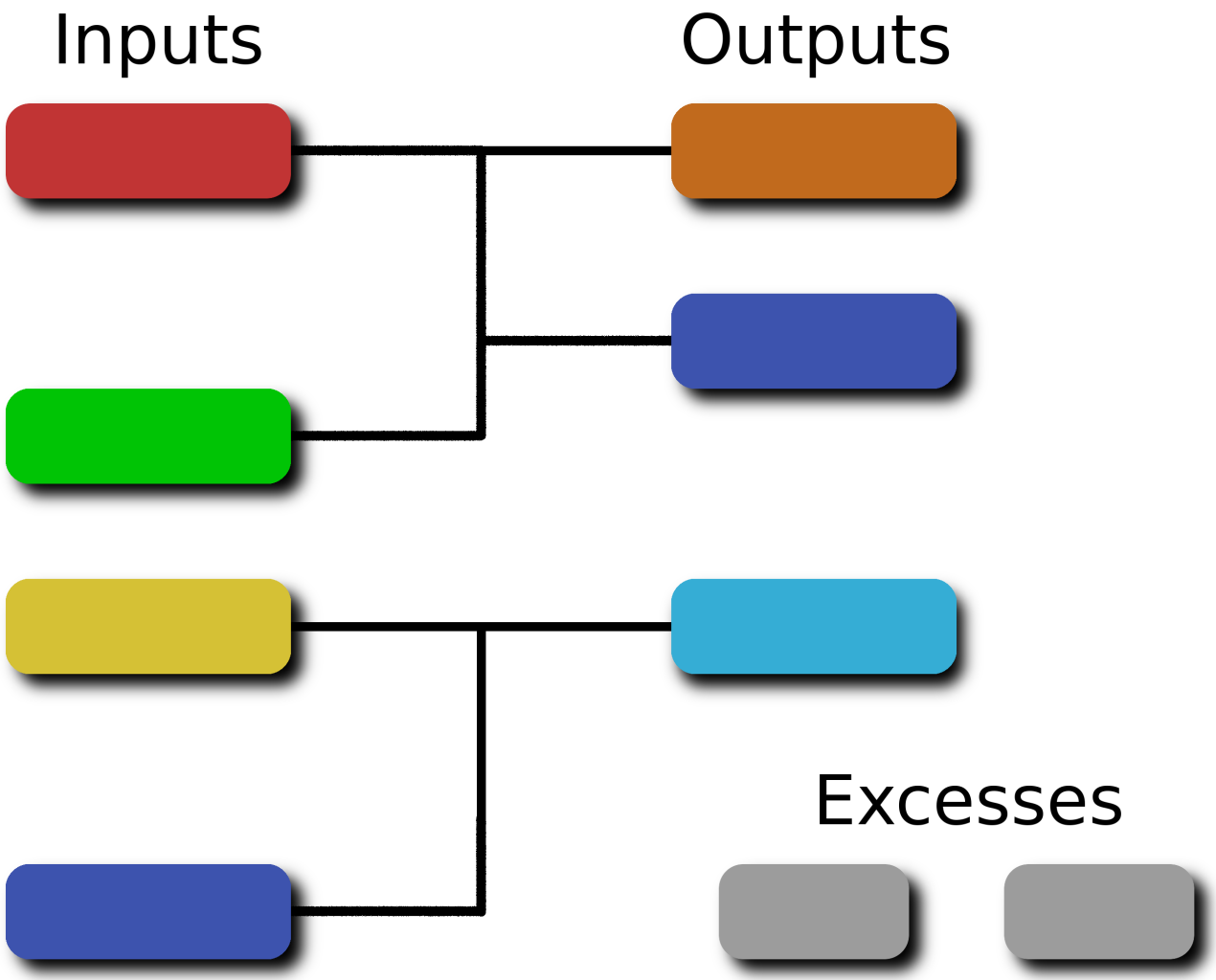
Mimblewimble Transactions



Mimblewimble Transactions



Mimblewimble Transactions



Mimblewimble Blocks

Blocks consist of:

- A merkle tree of transaction inputs.

Mimblewimble Blocks

Blocks consist of:

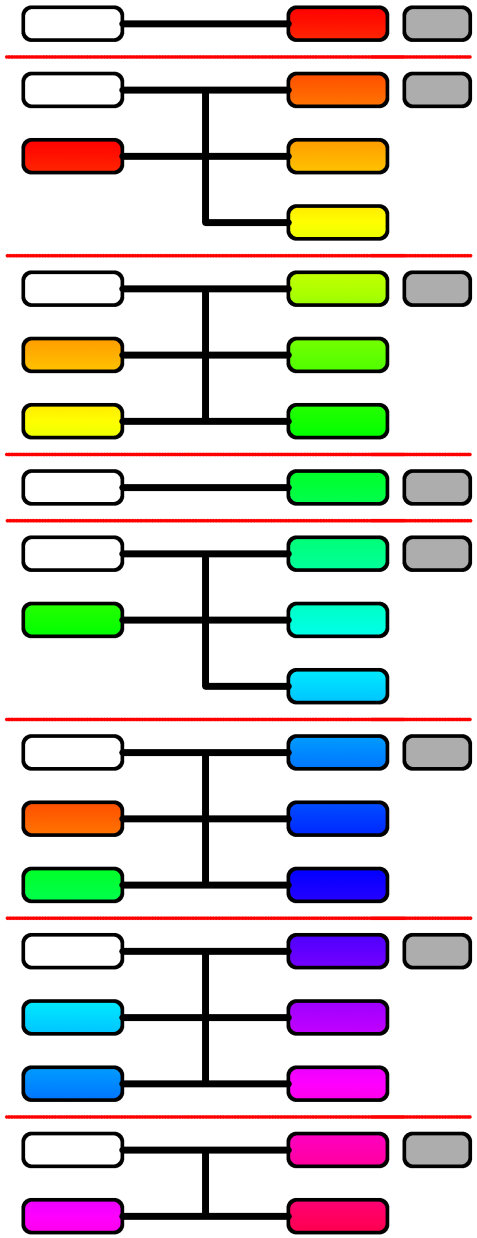
- A merkle tree of transaction inputs.
- A merkle tree of transaction outputs and rangeproofs.

Mimblewimble Blocks

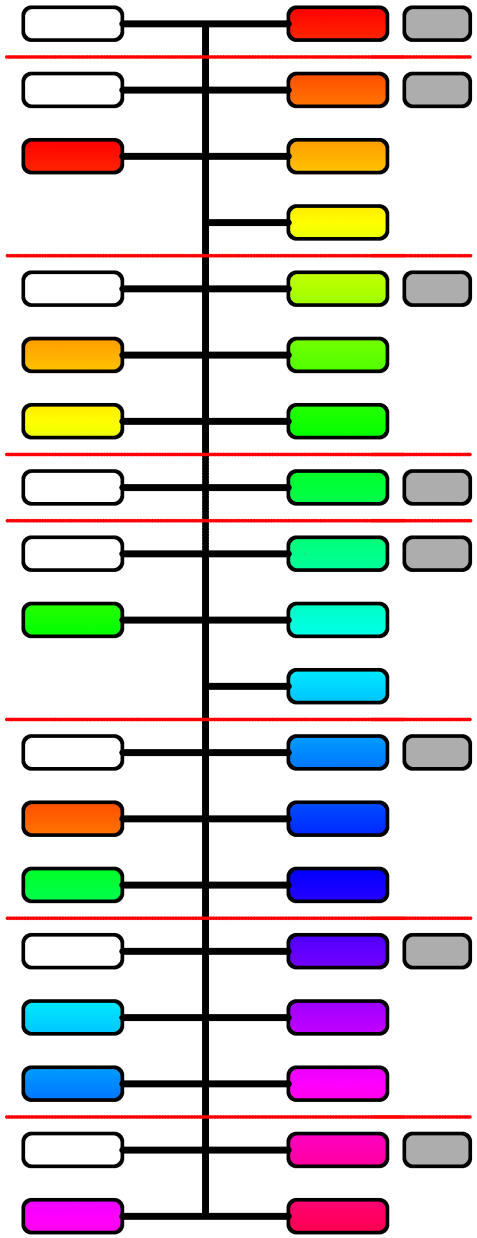
Blocks consist of:

- A merkle tree of transaction inputs.
- A merkle tree of transaction outputs and rangeproofs.
- A list of excess value(s) and signature(s)

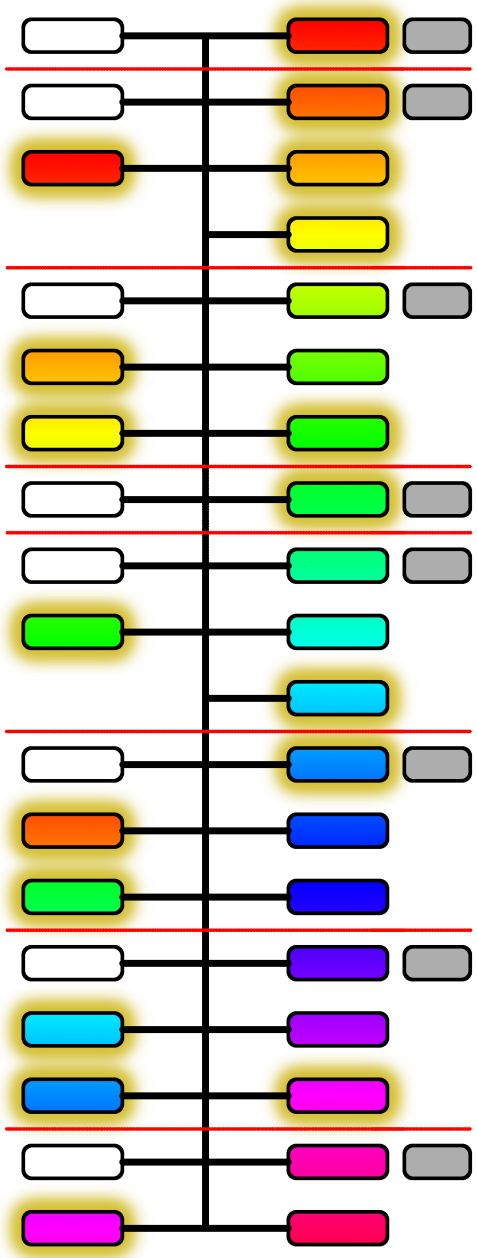
Mimblewimble Transactions



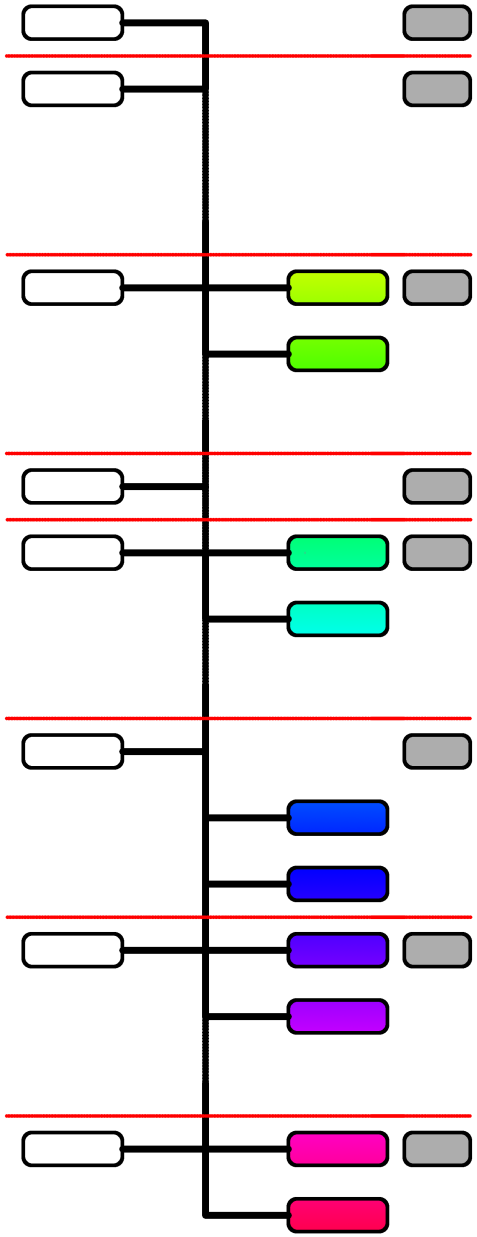
Mimblewimble Transactions



Mimblewimble Transactions



Mimblewimble Transactions



Trust Model: Transactions

A transaction is valid if:

- It is non-inflationary (total input amount equals total output amount)

Trust Model: Transactions

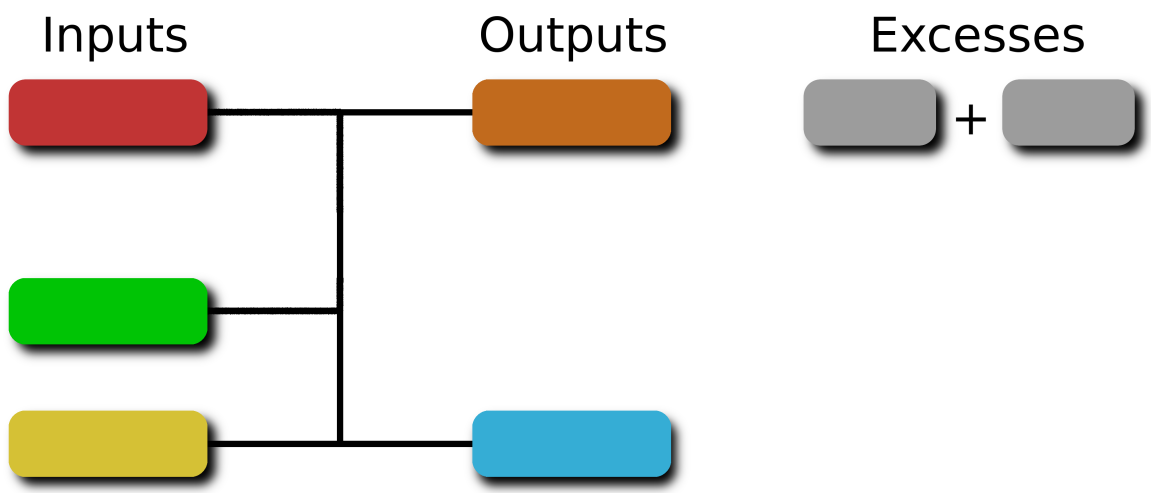
A transaction is valid if:

- It is non-inflationary (total input amount equals total output amount)
- The owner of the input(s) has signed off on it.

Trust Model: Transactions

A transaction is valid if:

- It is non-inflationary (total input amount equals total output amount)
- The owner of the input(s) has signed off on it.



Trust Model: Blockchain

It should be verifiable that

- A transaction, once committed to a block, cannot be reversed without doing enough work to rewrite the block (and all its descendants).

Trust Model: Blockchain

It should be verifiable that

- A transaction, once committed to a block, cannot be reversed without doing enough work to rewrite the block (and all its descendants).
- The current state of all coins reflects zero net theft and inflation.

Trust Model: Blockchain

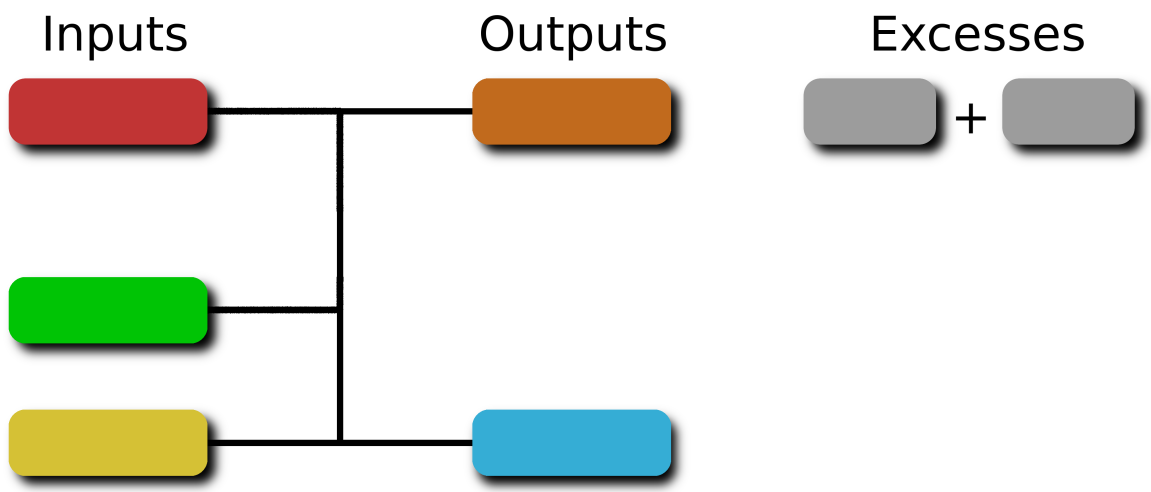
It should be verifiable that

- A transaction, once committed to a block, cannot be reversed without doing enough work to rewrite the block (and all its descendants).
- The current state of all coins reflects zero net theft and inflation.
- *The exact historical sequence of transactions does not need to be publicly verifable.*

Trust Model: Blockchain

It should be verifiable that

- A transaction, once committed to a block, cannot be reversed without doing enough work to rewrite the block (and all its descendants).
- The current state of all coins reflects zero net theft and inflation.
- *The exact historical sequence of transactions does not need to be publicly verifable.*



Trust Model: Block Verification

It is possible to verify the blockchain with only the following data:

- Block headers

Trust Model: Block Verification

It is possible to verify the blockchain with only the following data:

- Block headers
- Unspent outputs from each block

Trust Model: Block Verification

It is possible to verify the blockchain with only the following data:

- Block headers
- Unspent outputs from each block
- Excess values and signatures.

Trust Model: Block Verification

It is possible to verify the blockchain with only the following data:

- Block headers
- Unspent outputs from each block
- Excess values and signatures.
- Rangeproofs for the above (witness data)

Trust Model: Block Verification

It is possible to verify the blockchain with only the following data:

- Block headers
- Unspent outputs from each block
- Excess values and signatures.
- Rangeproofs for the above (witness data)
- Full blocks near the tip should be kept to handle reorgs

Trust Model: Block Verification

It is possible to verify the blockchain with only the following data:

- Block headers
- Unspent outputs from each block
- Excess values and signatures.
- Rangeproofs for the above (witness data)
- Full blocks near the tip should be kept to handle reorgs
- In Bitcoin there are 150 million transactions and 40 million unsigned transaction outputs: 21.6Gb of historic data, 2Gb of UTXOs and 100Gb of UTXO rangeproofs.

Sinking Signatures

Open problem 2 from the Voldemort paper: can we aggregate excess signatures?

- Difficult to do without also making them negatable in later blocks. (If you can add you can subtract.)

Sinking Signatures

Open problem 2 from the Voldemort paper: can we aggregate excess signatures?

- Difficult to do without also making them negatable in later blocks. (If you can add you can subtract.)
- We can by using pairings and (a variant of) Boneh-Lynn-Shacham signatures which sign *the current block height*.

Sinking Signatures

Open problem 2 from the Voldemort paper: can we aggregate excess signatures?

- Difficult to do without also making them negatable in later blocks. (If you can add you can subtract.)
- We can by using pairings and (a variant of) Boneh-Lynn-Shacham signatures which sign *the current block height*.
- Now only one excess and (multi-)signature per block.

Sinking Signatures

Open problem 2 from the Voldemort paper: can we aggregate excess signatures?

- Difficult to do without also making them negatable in later blocks. (If you can add you can subtract.)
- We can by using pairings and (a variant of) Boneh-Lynn-Shacham signatures which sign *the current block height*.
- Now only one excess and (multi-)signature per block.
- In Bitcoin there are 450000 blocks: so 22Mb of historic data (but 450k pairings to verify), 2Gb of UTXO and 100Gb of rangeproofs.

Sinking Signatures

What if we could combine blocks as well?

- It is possible to compress a chain of blockheaders to logarithmic size while still having a proof that takes the same expected work as the original [Back et. al. 2014; Kiayias, Lamprou, Stouka 2016].

Sinking Signatures

What if we could combine blocks as well?

- It is possible to compress a chain of blockheaders to logarithmic size while still having a proof that takes the same expected work as the original [Back et. al. 2014; Kiayias, Lamprou, Stouka 2016].
- Treat the blockchain as a "skiplist" where blocks have distant parents.

Sinking Signatures

What if we could combine blocks as well?

- It is possible to compress a chain of blockheaders to logarithmic size while still having a proof that takes the same expected work as the original [Back et. al. 2014; Kiayias, Lamprou, Stouka 2016].
- Treat the blockchain as a "skiplist" where blocks have distant parents.
- If excess values signed the *current height* and *previous heights*, the excesses and signatures can be combined for skipped blocks.

Sinking Signatures

What if we could combine blocks as well?

- It is possible to compress a chain of blockheaders to logarithmic size while still having a proof that takes the same expected work as the original [Back et. al. 2014; Kiayias, Lamprou, Stouka 2016].
- Treat the blockchain as a "skiplist" where blocks have distant parents.
- If excess values signed the *current height* and *previous heights*, the excesses and signatures can be combined for skipped blocks.
- This is a *sinking signature*, which can be made log-sized. [Poelstra, Kulkarni 2016]

Sinking Signatures

What if we could combine blocks as well?

- It is possible to compress a chain of blockheaders to logarithmic size while still having a proof that takes the same expected work as the original [Back et. al. 2014; Kiayias, Lamprou, Stouka 2016].
- Treat the blockchain as a "skiplist" where blocks have distant parents.
- If excess values signed the *current height* and *previous heights*, the excesses and signatures can be combined for skipped blocks.
- This is a *sinking signature*, which can be made log-sized. [Poelstra, Kulkarni 2016]
- Simulations show a 500000-block chain can likely be compressed to 300 blocks. We now have *1Mb* of historic data (20 seconds on one core to verify), 2Gb of UTXO and 100Gb of rangeproofs.

Trust Model: Blockchain verifitation, redox

- Compact chains have same *expected work* to produce but much higher variance than the original chain.

Trust Model: Blockchain verifitation, redox

- Compact chains have same *expected work* to produce but much higher variance than the original chain.
- With nontrivial probability a compact chain can be produced in less work than one block; thus compact chains prove no work.

Trust Model: Blockchain verifitation, redox

- Compact chains have same *expected work* to produce but much higher variance than the original chain.
- With nontrivial probability a compact chain can be produced in less work than one block; thus compact chains prove no work.
- We thus distinguish between *expected work* which affects economic incentives for forgeries, and *proven work* which assures a verifier an historic fact about how a chain was produced.

Trust Model: Blockchain verifitation, redox

- Compact chains have same *expected work* to produce but much higher variance than the original chain.
- With nontrivial probability a compact chain can be produced in less work than one block; thus compact chains prove no work.
- We thus distinguish between *expected work* which affects economic incentives for forgeries, and *proven work* which assures a verifier an historic fact about how a chain was produced.
- Mimblewimble verifiers should demand a month or two of non-compact blocks on top of a compact chain. This proves a month or two of work, but a forger expects to do as much work as the entire chain. So no "incentive cliff" and verifiers can also be assured that their chain was no accident.

Next Steps

- Development, development, development!

Next Steps

- Development, development, development!
- Nail down chain parameters, in particular choice of pairing-friendly curve.

Next Steps

- Development, development, development!
- Nail down chain parameters, in particular choice of pairing-friendly curve.
- Sidechain?

Open Problems

- Smaller rangeproofs? Aggregation of rangeproofs?

Open Problems

- Smaller rangeproofs? Aggregation of rangeproofs?
- Peer-to-peer protocol that can handle transaction merging

Open Problems

- Smaller rangeproofs? Aggregation of rangeproofs?
- Peer-to-peer protocol that can handle transaction merging
- Quantum resistance

Thank You

Andrew Poelstra <grindelwald@wpsoftware.net>