Mimblewimble: Private, Massively-Prunable Blockchains

Andrew Poelstra

grindelwald@wpsoftware.net

January 27, 2016

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.
- 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?

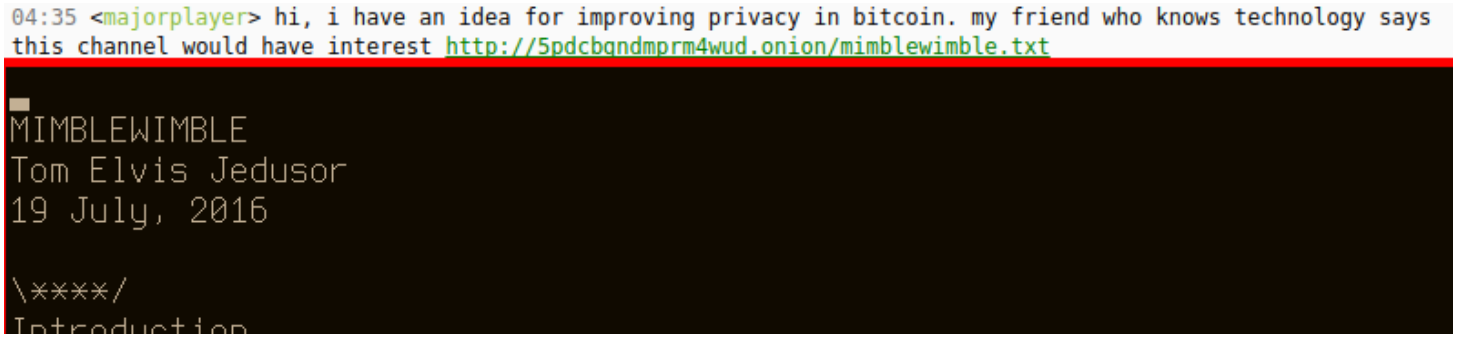
- Mimblewimble is a design for a blockchain-based ledger that is very different from Bitcoin.
- In Bitcoin transactions, old outputs sign new outputs; outputs have "script pubkeys" that are independent of each other. In Mimblewimble transactions, outputs have only EC pubkeys, and the difference between new outputs' keys and old ones' is multisigned by all transacting parties.
- These outputs, or "transaction kernels", are the only thing that needs to be retained in the blockchain.

What is Mimblewimble?

- Mimblewimble is a design for a blockchain-based ledger that is very different from Bitcoin.
- 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.
- These outputs, or "transaction kernels", are the only thing that needs to be retained in the blockchain.
- Mimblewimble outputs (and inputs) are inherently scriptless.

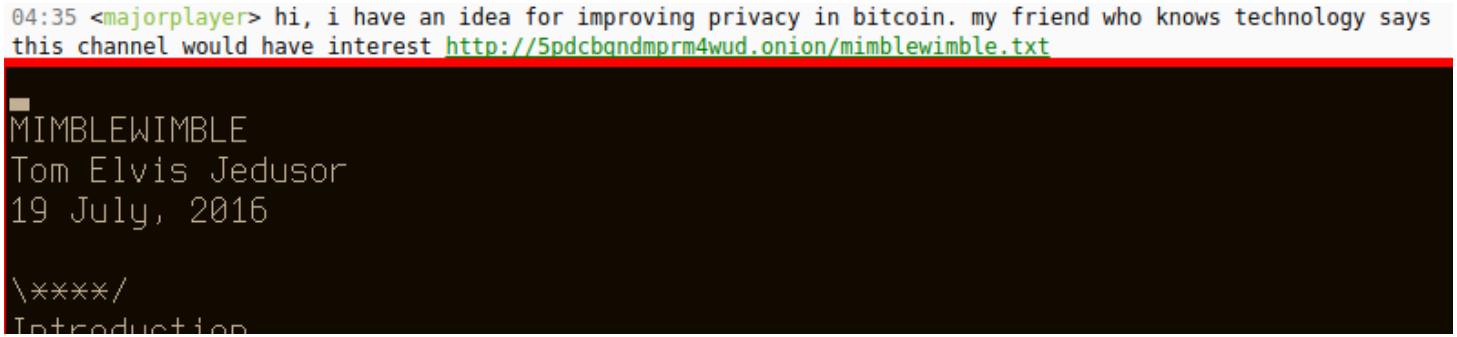
History

- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC and disappears



History

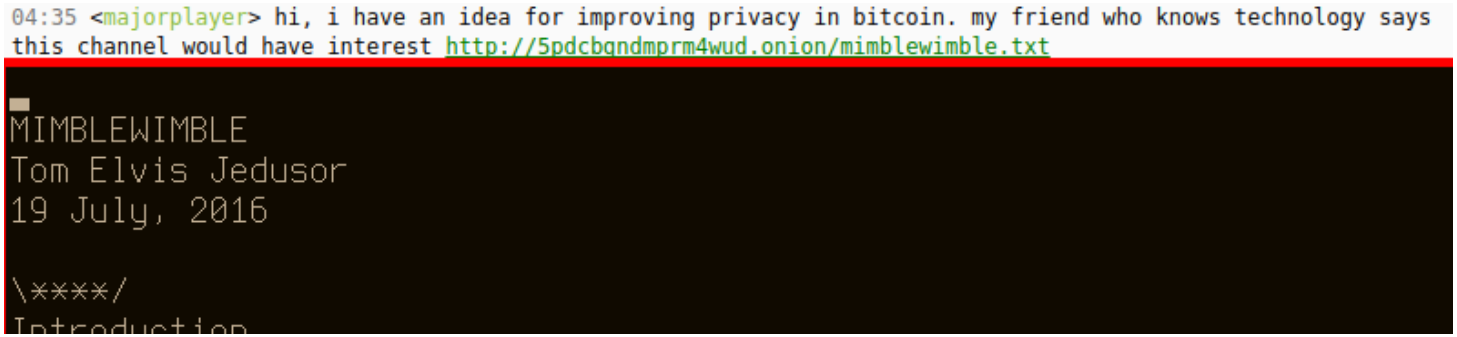
- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC and disappears



- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.

History

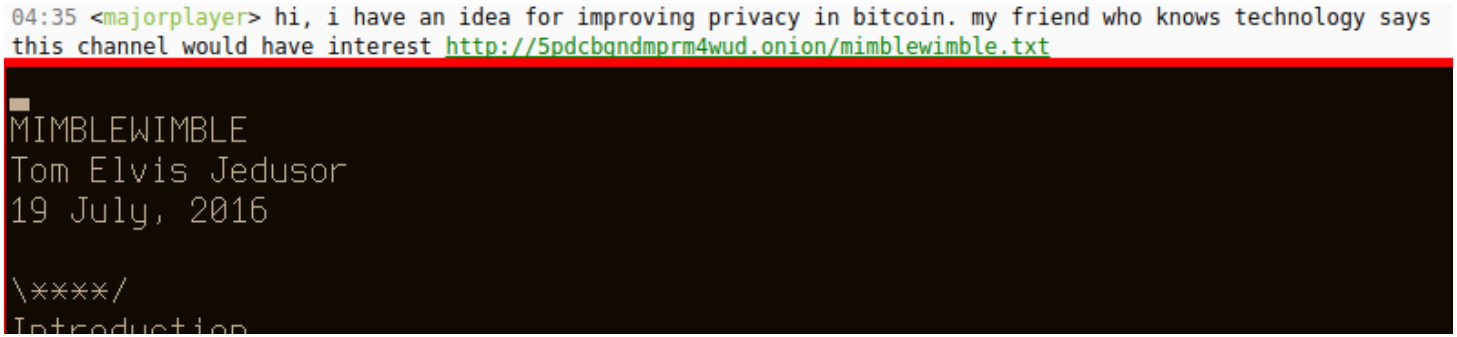
- 04:30 UTC, August 2nd, 2016: "Tom Elvis Jedusor" posts a .onion link to a text file on IRC and disappears



- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.
- 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 and disappears



- Next morning: myself and Bryan Bishop verify it's actually just text and rehost it.
- 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.
- October 8th: paper shows Avi Kularni's and my work extending/formalizing this; presented at Scaling Bitcoin Milan

History

- At 23:47 UTC, October 20, “Ignotus Peverell” appeared on IRC announcing a project to implement MimbleWimble.

History

- At 23:47 UTC, October 20, “Ignotus Peverell” appeared on IRC announcing a project to implement MimbleWimble.
- A few minutes later, Bryan Bishop called me to tell me to join the conversation. I pointed out that aggregate signatures give space savings on top of the Voldemort scheme, even without new crypto.

History

- At 23:47 UTC, October 20, "Ignotus Peverell" appeared on IRC announcing a project to implement MimbleWimble.
- A few minutes later, Bryan Bishop called me to tell me to join the conversation. I pointed out that aggregate signatures give space savings on top of the Voldemort scheme, even without new crypto.
- Other Harry Potter characters arrived over the next few weeks; the project continues to move forward. Though I've been involved with the project, I have not contributed any code. I am certainly not Ignotus Peverell.

History

- At 23:47 UTC, October 20, "Ignotus Peverell" appeared on IRC announcing a project to implement MimbleWimble.
- A few minutes later, Bryan Bishop called me to tell me to join the conversation. I pointed out that aggregate signatures give space savings on top of the Voldemort scheme, even without new crypto.
- Other Harry Potter characters arrived over the next few weeks; the project continues to move forward. Though I've been involved with the project, I have not contributed any code. I am certainly not Ignotus Peverell.
- January 17th, 2017: I meet with Ethan Heilman of TumbleBit fame. We go back and forth on Lightning, ZKCP, etc., and discover a powerful new primitive to get all these things on MimbleWimble.

History

- At 23:47 UTC, October 20, "Ignotus Peverell" appeared on IRC announcing a project to implement MimbleWimble.
- A few minutes later, Bryan Bishop called me to tell me to join the conversation. I pointed out that aggregate signatures give space savings on top of the Voldemort scheme, even without new crypto.
- Other Harry Potter characters arrived over the next few weeks; the project continues to move forward. Though I've been involved with the project, I have not contributed any code. I am certainly not Ignotus Peverell.
- January 17th, 2017: I meet with Ethan Heilman of TumbleBit fame. We go back and forth on Lightning, ZKCP, etc., and discover a powerful new primitive to get all these things on MimbleWimble.
- The next day, Ruben Somsen messaged me on Reddit explaining how to get non-expiring bidirectonal channels.

Mimblewimble Transactions

A Mimblewimble transaction is the following data:

- Inputs (references to old outputs).

Mimblewimble Transactions

A Mimblewimble transaction is the following data:

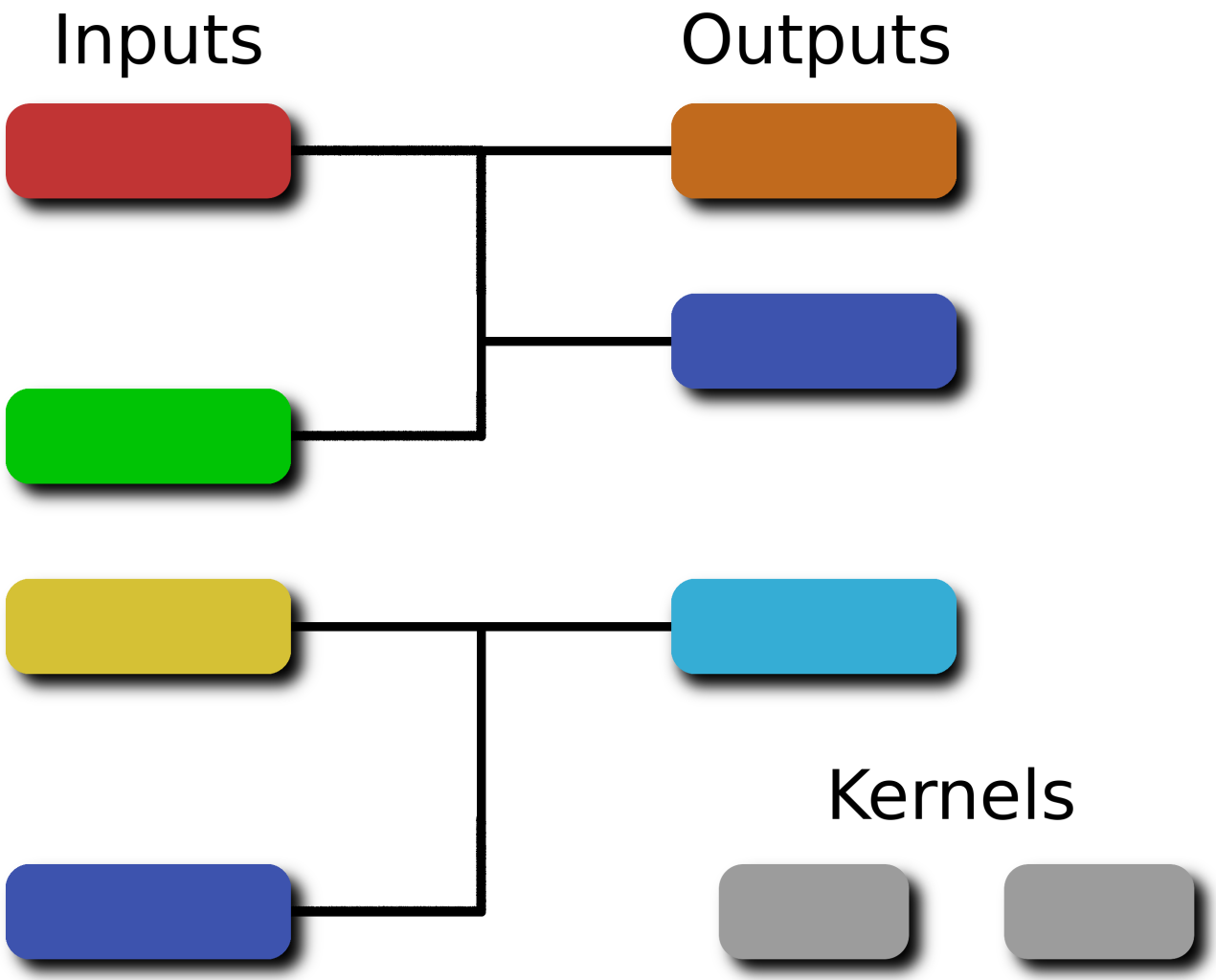
- Inputs (references to old outputs).
- Outputs: confidential transaction outputs (group elements, which blind and commit to amounts), plus rangeproofs.

Mimblewimble Transactions

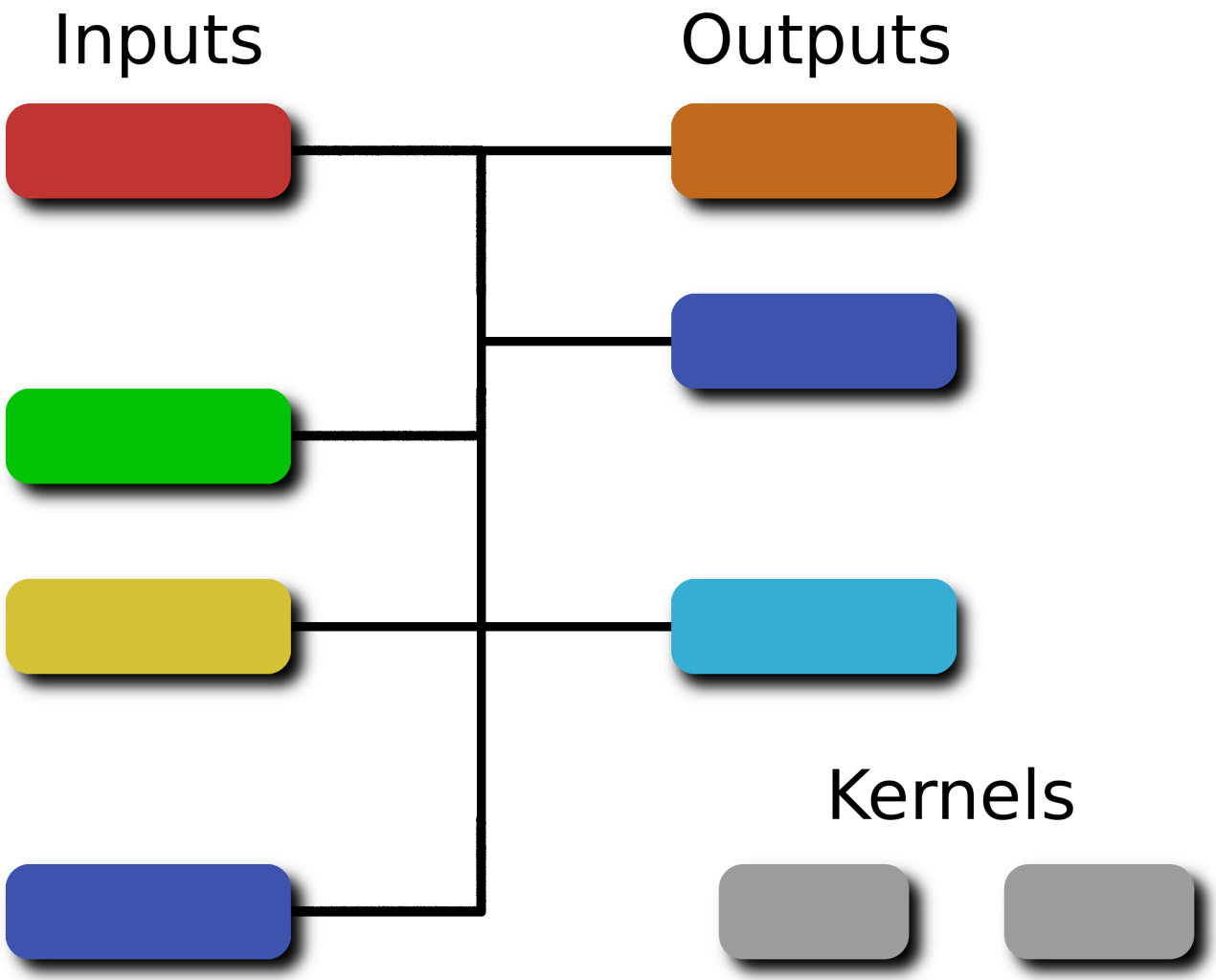
A Mimblewimble transaction is the following data:

- Inputs (references to old outputs).
- Outputs: confidential transaction outputs (group elements, which blind and commit to amounts), plus rangeproofs.
- Kernel: algebraically, difference between outputs and inputs (group element); morally a multisignature key for all transacting parties.
- Kernel signature: the kernel must sign itself to prove that the transaction is honestly constructed; by signing other blockchain-enforced data we can add additional functionality (e.g. locktimes).

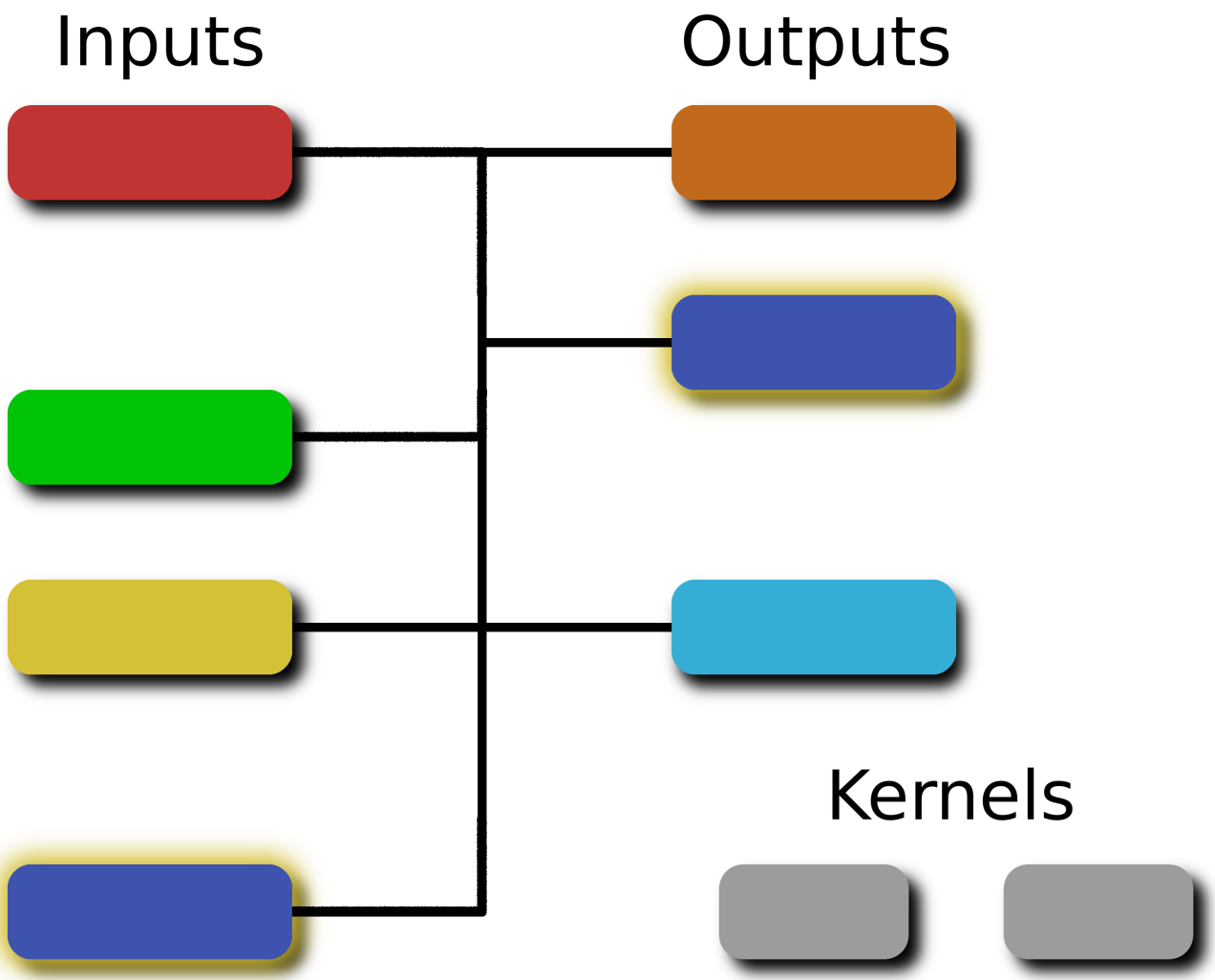
Mimblewimble Transactions



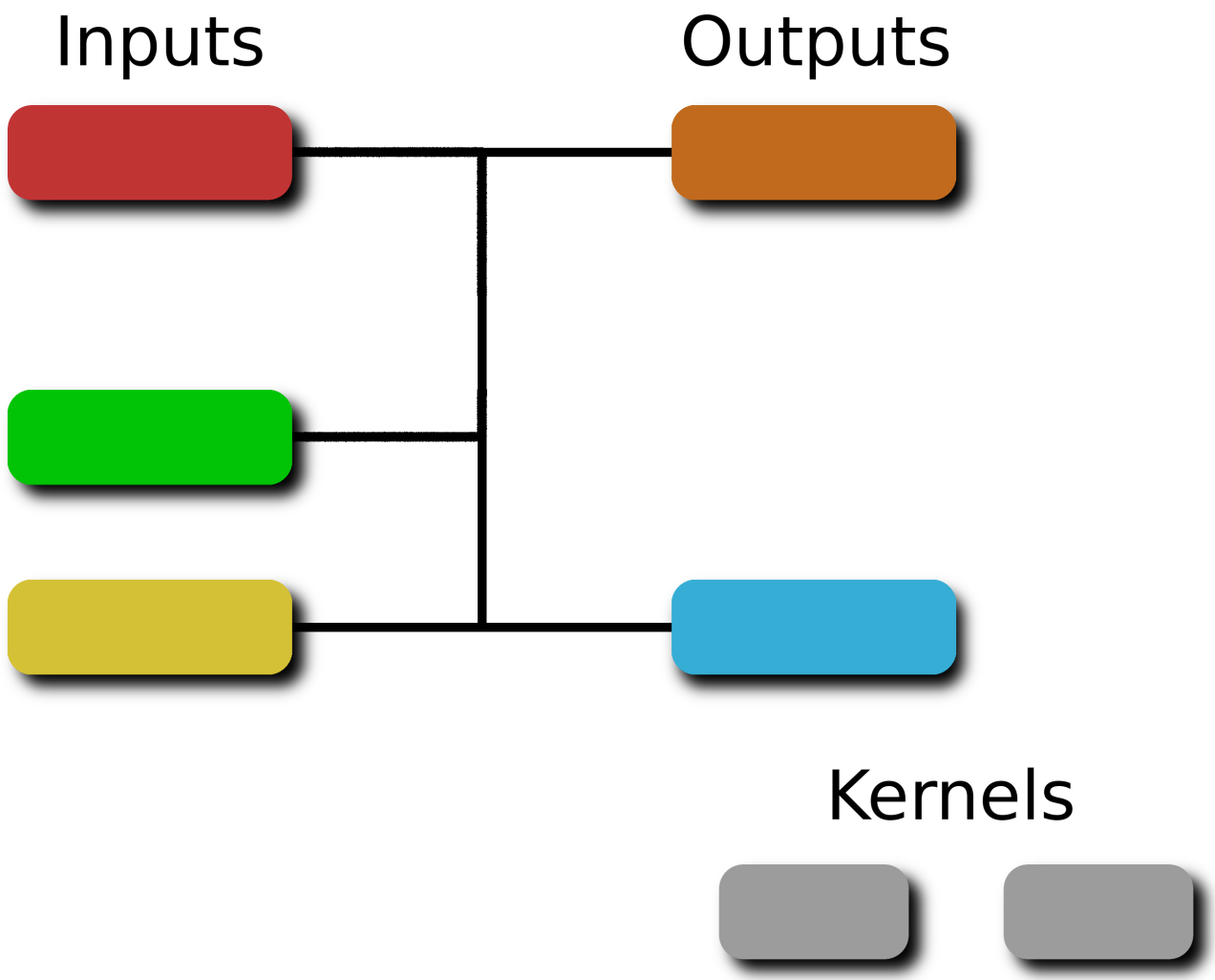
Mimblewimble Transactions



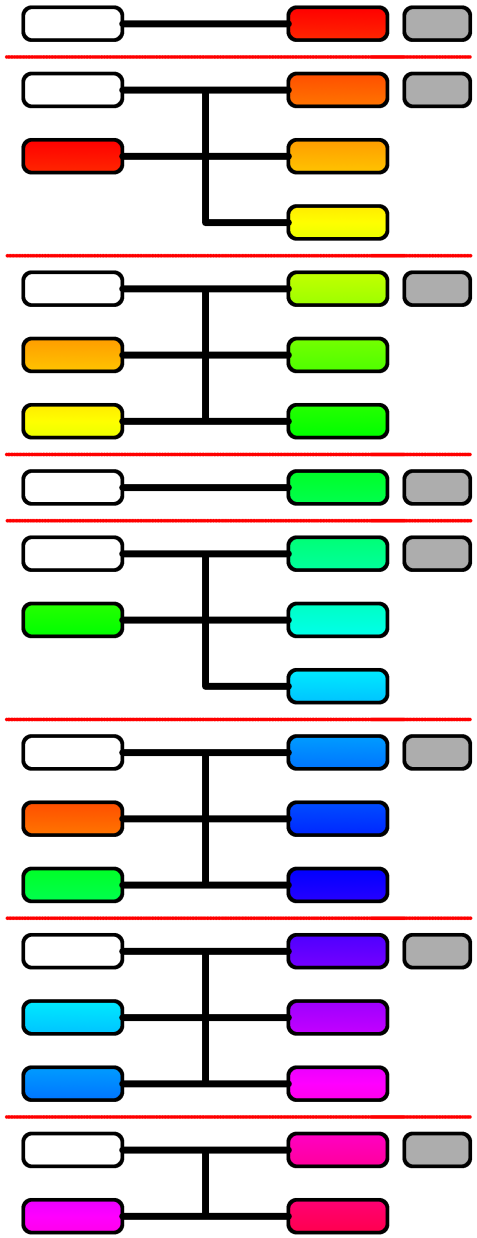
Mimblewimble Transactions



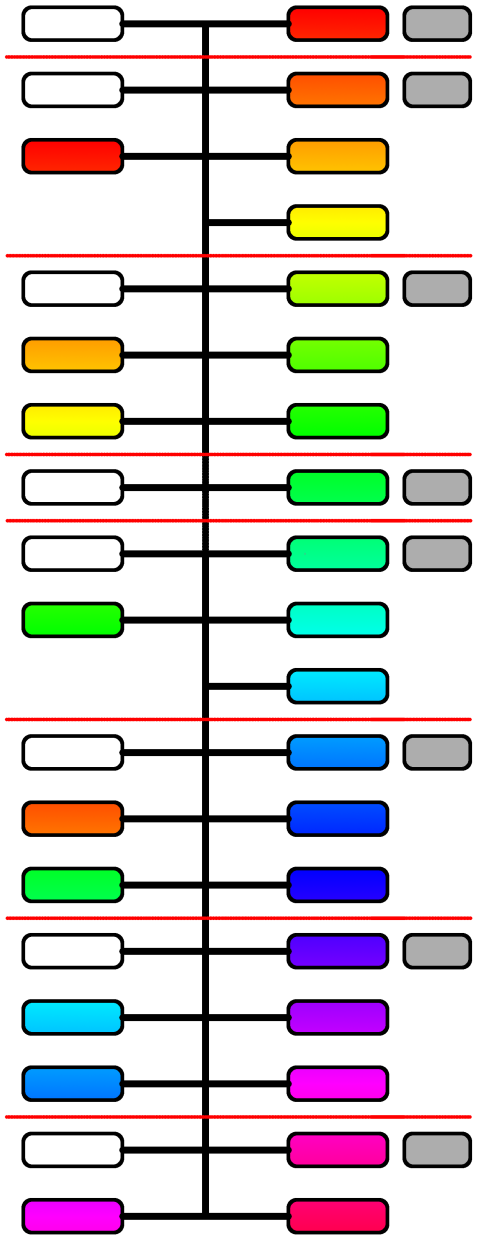
Mimblewimble Transactions



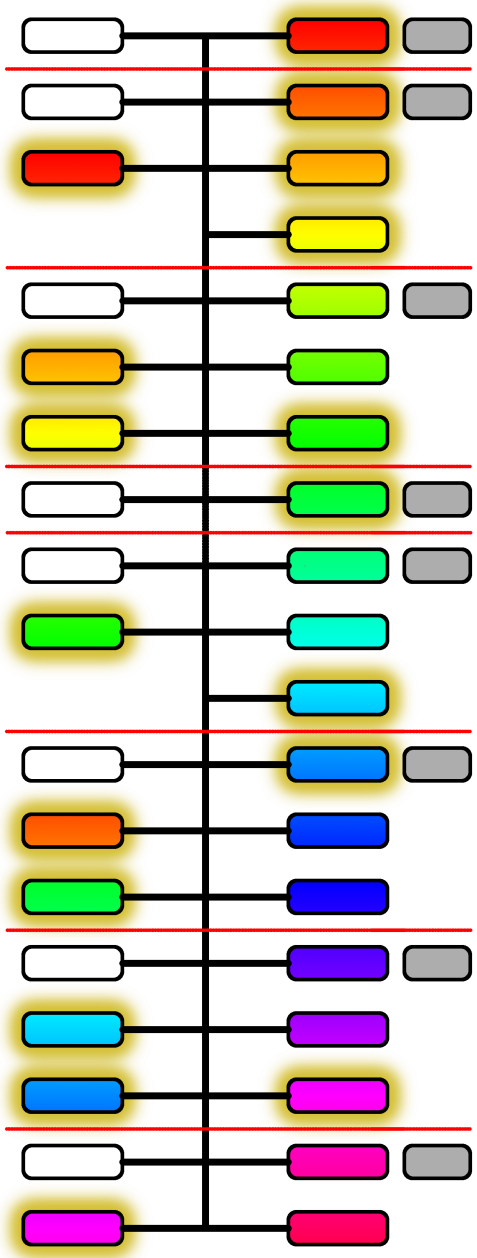
Mimblewimble Transactions



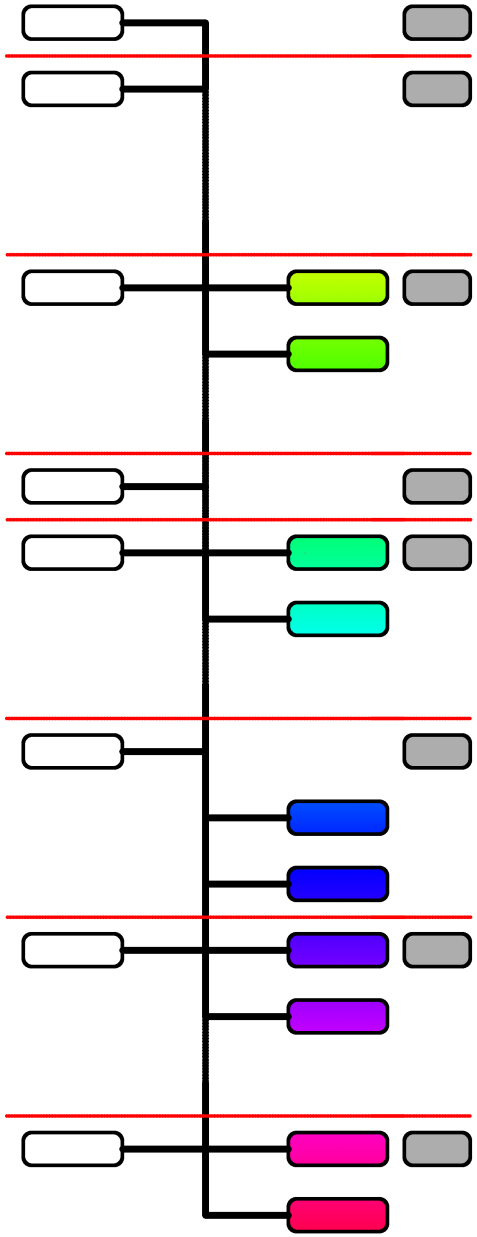
Mimblewimble Transactions



Mimblewimble Transactions



Mimblewimble Transactions



Scaling: Real Numbers

- In Bitcoin there are 150 million transactions with about 350 million outputs, and 45 million of which are unspent.

Scaling: Real Numbers

- In Bitcoin there are 150 million transactions with about 350 million outputs, and 45 million of which are unspent.
- This takes about 100Gb of space on disk today; with CT this would be over 1Tb!

Scaling: Real Numbers

- In Bitcoin there are 150 million transactions with about 350 million outputs, and 45 million of which are unspent.
- This takes about 100Gb of space on disk today; with CT this would be over 1Tb!
- MimbleWimble gives us CT and requires storing: 15Gb of transaction kernels, headers etc.; 2Gb of unspent outputs, and 100Gb of UTXO rangeproofs.

Scaling: Real Numbers

- In Bitcoin there are 150 million transactions with about 350 million outputs, and 45 million of which are unspent.
- This takes about 100Gb of space on disk today; with CT this would be over 1Tb!
- MimbleWimble gives us CT and requires storing: 15Gb of transaction kernels, headers etc.; 2Gb of unspent outputs, and 100Gb of UTXO rangeproofs.
- In pre-segwit Bitcoin, none of this is separable "witness data" which can be dropped in exchange for trust. In MW the rangeproofs are, leaving less than 20Gb of normative blockchain space.

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 structure of each individual transaction does not need to be publicly verifiable.*

Claim or Refund

- MimbleWimble supports Information $\Leftrightarrow$ Money

Claim or Refund

- MimbleWimble supports Information $\Leftrightarrow$ Money
- Kernels sign not only themselves; but also (optionally) a locktime and a hash. A valid transaction must include the preimage of this hash.

Claim or Refund

- MimbleWimble supports Information $\Leftrightarrow$ Money
- Kernels sign not only themselves; but also (optionally) a locktime and a hash. A valid transaction must include the preimage of this hash.
- To do a hash-locktimed transaction, buying party sends coins to a 2-of-2 multisig output, conditioned on the seller signing a transaction to return the money at a later block.

Claim or Refund

- MimbleWimble supports Information $\Leftrightarrow$ Money
- Kernels sign not only themselves; but also (optionally) a locktime and a hash. A valid transaction must include the preimage of this hash.
- To do a hash-locktimed transaction, buying party sends coins to a 2-of-2 multisig output, conditioned on the seller signing a transaction to return the money at a later block.
- The buyer then signs a transaction sending the money to the seller, signing the hash she wants the preimage to.

Claim or Refund

- MimbleWimble supports Information $\Leftrightarrow$ Money
- Kernels sign not only themselves; but also (optionally) a locktime and a hash. A valid transaction must include the preimage of this hash.
- To do a hash-locktimed transaction, buying party sends coins to a 2-of-2 multisig output, conditioned on the seller signing a transaction to return the money at a later block.
- The buyer then signs a transaction sending the money to the seller, signing the hash she wants the preimage to.
- The seller, to complete the transaction and take the coins, must reveal the preimage.

Claim or Refund

- MimbleWimble supports Information $\Leftrightarrow$ Money
- Kernels sign not only themselves; but also (optionally) a locktime and a hash. A valid transaction must include the preimage of this hash.
- To do a hash-locktimed transaction, buying party sends coins to a 2-of-2 multisig output, conditioned on the seller signing a transaction to return the money at a later block.
- The buyer then signs a transaction sending the money to the seller, signing the hash she wants the preimage to.
- The seller, to complete the transaction and take the coins, must reveal the preimage.
- This primitive is the basis of: cross-chain atomic swaps, ZKCP's, Lighting Channels, TumbleBit, and more.

Secret Atomic Swaps

- For atomic swaps, we are exchanging a signature on one transaction for a signature on another.

Secret Atomic Swaps

- For atomic swaps, we are exchanging a signature on one transaction for a signature on another.
- This can be done algebraically such that the two signatures are not related in a publicly verifiable way (and is deniable)

Secret Atomic Swaps

- For atomic swaps, we are exchanging a signature on one transaction for a signature on another.
- This can be done algebraically such that the two signatures are not related in a publicly verifiable way (and is deniable)
- Since the locktimed transaction never touches the blockchain unless something goes wrong, the default case is that the atomic swap is indistinguishable from any other transaction.

Next Steps

- Development, development, development!

Next Steps

- Development, development, development!
- Wallet support: multisig rangeproofs, triggers, secret atomic swaps, etc.

Next Steps

- Development, development, development!
- Wallet support: multisig rangeproofs, triggers, secret atomic swaps, etc.
- ValueShuffle

Open Problems

- Smaller rangeproofs? Aggregation of rangeproofs?

Open Problems

- Smaller rangeproofs? Aggregation of rangeproofs?
- Peer-to-peer layer that avoids monitoring (ValueShuffle?)

Open Problems

- Smaller rangeproofs? Aggregation of rangeproofs?
- Peer-to-peer layer that avoids monitoring (ValueShuffle?)
- Quantum resistance

Thank You

Andrew Poelstra <grindelwald@wpsoftware.net>