MIMBLEWIMBLE
Tom Elvis Jedusor
19 July, 2016

****/
Introduction
/****\

Bitcoin is the first widely used financial system for which all the necessary data to validate the system status can be cryptographically verified by anyone. However, it accomplishes this feat by storing all transactions in a public database called "the blockchain" and someone who genuinely wishes to check this state must download the whole thing and basically replay each transaction, check each one as they go. Meanwhile, most of these transactions have not affected the actual final state (they create outputs that are destroyed a transaction later).

At the time of this writing, there were nearly 150 million transactions committed in the blockchain, which must be replayed to produce a set of only 4 million unspent outputs.

It would be better if an auditor needed only to check data on the outputs themselves, but this is impossible because they are valid if and only if the output is at the end of a chain of previous outputs, each signs the next. In other words, the whole blockchain must be validated to confirm the final state.

Add to this that these transactions are cryptographically atomic, it is clear what outputs go into every transaction and what emerges. The "transaction graph" resulting reveals a lot of information and is subjected to analysis by many companies whose business model is to monitor and control the lower classes. This makes it very non-private and even dangerous for people to use.

Some solutions to this have been proposed. Greg Maxwell discovered to encrypt the amounts, so that the graph of the transaction is faceless but still allow validation that the sums are correct [1]. Dr Maxwell also produced CoinJoin, a system for Bitcoin users to combine interactively transactions, confusing the transaction graph. Nicolas van Saberhagen has developed a system to blind the transaction entries, goes much further to cloud the transaction graph (as well as not needed the user interaction) [3]. Later, Shen Noether combined the two approaches to obtain "confidential transactions" of Maxwell AND the darkening of van Saberhagen [4].

These solutions are very good and would make Bitcoin very safe to use. But the problem of too much data is made even worse. Confidential transactions require multi-kilobyte proofs on every output, and van Saberhagen signatures require every output to be stored for ever, since it is not possible to tell when they are truly spent.

Dr. Maxwell's CoinJoin has the problem of needing interactivity. Dr. Yuan Horas Mouton fixed this by making transactions freely mergeable [5], but he needed to use pairing-based cryptography, which is potentially slower and more difficult to trust. He called this "one-way aggregate signatures" (OWAS).

OWAS had the good idea to combine the transactions in blocks. Imagine that we can combine across blocks (perhaps with some glue data) so that when the outputs are created and destroyed, it is the same as if they never existed. Then, to validate the entire chain, users only need to know when money is entered into the system (new money in each block as in Bitcoin or Monero or peg-ins for sidechains [6]) and final unspent outputs, the rest can be removed and forgotten. Then we can have Confidential Transactions to hide the amounts and OWAS to blur the transaction graph, and use LESS space than Bitcoin to allow users to fully verify the blockchain. And also imagine that we must not pairing-based cryptography or new hypotheses, just regular discrete logarithms signatures like Bitcoin. Here is what I propose.

I call my creation Mimblewimble because it is used to prevent the blockchain from

talking about all user's information [7].

****/
Confidential Transactions and OWAS
/****\

The first thing we need to do is remove Bitcoin Script. This is sad, but it is too powerful so it is impossible to merge transactions using general scripts. We will demonstrate that confidential transactions of Dr. Maxwell are enough (after some small modification) to authorize spending of outputs and also allows to make combined transactions without interaction. This is in fact identical to OWAS, and allows relaying nodes take some transaction fee or the recipient to change the transaction fees. These additional things Bitcoin can not do, we get for free.

We start by reminding the reader how confidential transactions work. First, the amounts are coded by the following equation:

    C = r*G + v*H

where C is a Pedersen commitment, G and H are fixed nothing-up-my-sleeve elliptic curve group generators, v is the amount, and r is a secret random blinding key.

Attached to this output is a rangeproof which proves that v is in [0, 2^64], so that user cannot exploit the blinding to produce overflow attacks, etc.

To validate a transaction, the verifer will add commitments for all outputs, plus f*H (f here is the transaction fee which is given explicitly) and subtracts all input commitments. The result must be 0, which proves that no amount was created or destroyed overall.

We note that to create such a transaction, the user must know the sum of all the values of r for commitments entries. Therefore, the r-values (and their sums) act as secret keys. If we can make the r output values known only to the recipient, then we have an authentication system! Unfortunately, if we keep the rule that commits all add to 0, this is impossible, because the sender knows the sum of all _his_ r values, and therefore knows the receipient's r values sum to the negative of that. So instead, we allow the transaction to sum to a nonzero value k*G, and require a signature of an empty string with this as key, to prove its amount component is zero.

We let transactions have as many k*G values as they want, each with a signature, and sum them during verification.

To create transactions sender and recipient do following ritual:

1. Sender and recipient agree on amount to be sent. Call this b.

2. Sender creates transaction with all inputs and change output(s), and gives recipient the total blinding factor (r-value of change minus r-values of inputs) along with this transaction. So the commitments sum to r*G - b*H.

3. Recipient chooses random r-values for his outputs, and values that sum to b minus fee, and adds these to transaction (including range proof). Now the commitments sum to k*G - fee*H for some k that only recipient knows.

4. Recipient attaches signature with k to the transaction, and the explicit fee. It has done.

Now, creating transactions in this manner supports OWAS already. To show this, suppose we have two transactions that have a surplus k1*G and k2*G, and the attached signatures with these. Then you can combine the lists of inputs and outputs of the two transactions, with both k1*G and k2*G to the mix, and voilÃ¡! is again a valid transaction. From the combination, it is impossible to say which outputs or inputs are from which original transaction.

Because of this, we change our block format from Bitcoin to this information:

1. Explicit amounts for new money (block subsidy or sidechain peg-ins) with

whatever else data this needs. For a sidechain peg-in maybe it references a Bitcoin transaction that commits to a specific excess k*G value?

2. Inputs of all transactions

3. Outputs of all transactions

4. Excess k*G values for all transactions

Each of these are grouped together because it do not matter what the transaction boundaries are originally. In addition, Lists 2 3 and 4 should be required to be coded in alphabetical order, since it is quick to check and prevents the block creator of leaking any information about the original transactions.

Note that the outputs are now identified by their hash, and not by their position in a transaction that could easily change. Therefore, it should be banned to have two unspent outputs are equal at the same time, to avoid confusion.

****/
Merging Transactions Across Blocks
/****\

Now, we have used Dr. Maxwell's Confidential Transactions to create a noninteractive version of Dr. Maxwell's CoinJoin, but we have not seen the last of marvelous Dr. Maxwell !
We need another idea, transaction cut-through, he described in [8]. Again, we create a noninteractive version of this, and to show how it is used with several blocks.

We can imagine now each block as one large transaction. To validate it, we add all the output commitments together, then subtracts all input commitments, k*G values, and all explicit input amounts times H. We find that we could combine transactions from two blocks, as we combined transactions to form a single block, and the result is again a valid transaction. Except now, some output commitments have an input commitment exactly equal to it, where the first block's output was spent in the second block. We could remove both commitments and still have a valid transaction. In fact, there is not even need to check the rangeproof of the deleted output.

The extension of this idea all the way from the genesis block to the latest block, we see that EVERY nonexplicit input is deleted along with its referenced output. What remains are only the unspent outputs, explicit input amounts and every k*G value. And this whole mess can be validated as if it were one transaction: add all unspent commitments output, subtract the values k*G, validate explicit input amounts (if there is anything to validate) then subtract them times H. If the sum is 0, the entire chain is good.

What is this mean? When a user starts up and downloads the chain he needs the following data from each block:

1. Explicit amounts for new money (block subsidy or sidechain peg-ins) with whatever else data this needs.

2. Unspent outputs of all transactions, along with a merkle proof that each output appeared in the original block.

3. Excess k*G values for all transactions.

Bitcoin today there are about 423000 blocks, totaling 80GB or so of data on the hard drive to validate everything. These data are about 150 million transactions and 5 million unspent nonconfidential outputs. Estimate how much space the number of transactions take on a Mimblewimble chain. Each unspent output is around 3Kb for rangeproof and Merkle proof. Each transaction also adds about 100 bytes: a k*G value and a signature. The block headers and explicit amounts are negligible. Add this together and get 30Gb -- with a confidential transaction and obscured transaction graph!

****/
Questions and Intuition
/****\

Here are some questions that since these weeks, dreams asked me and I woke up sweating. But in fact it is OK.

Q. If you delete the transaction outputs, user cannot verify the rangeproof and maybe a negative amount is created.

A. This is OK. For the entire transaction to validate all negative amounts must have been destroyed. User have SPV security only that no illegal inflation happened in the past, but the user knows that _at this time_ no inflation occurred.

Q. If you delete the inputs, double spending can happen.

A. In fact, this means: maybe someone claims that some unspent output was spent in the old days. But this is impossible, otherwise the sum of the combined transaction could not be zero.

An exception is that if the outputs are amount zero, it is possible to make two that are negatives of each other, and the pair can be revived without anything breaks. So to prevent consensus problems, outputs 0-amount should be banned. Just add H at each output, now they all amount to at least 1.

****/
Future Research
/****\

Here are some questions I can not answer at the time of this writing.

1. What script support is possible? We would need to translate script operations into some sort of discrete logarithm information.

2. We require user to check all k*G values, when in fact all that is needed is that their sum is of the form k*G. Instead of using signatures is there another proof of discrete logarithm that could be combined?

3. There is a denial-of-service option when a user downloads the chain, the peer can give gigabytes of data and list the wrong unspent outputs. The user will see that the result do not add up to 0, but cannot tell where the problem is.

   For now maybe the user should just download the blockchain from a Torrent or something where the data is shared between many users and is reasonably likely to be correct.

[1] https://people.xiph.org/~greg/confidential_values.txt
[2] https://bitcointalk.org/index.php?topic=279249.0
[3] https://cryptonote.org/whitepaper.pdf
[4] https://eprint.iacr.org/2015/1098.pdf
[5] https://download.wpsoftware.net/bitcoin/wizardry/horasyuanmouton-owas.pdf
[6] http://blockstream.com/sidechains.pdf
[7] http://fr.harrypotter.wikia.com/wiki/SortilÃ¨ge_de_Langue_de_Plomb
[8] https://bitcointalk.org/index.php?topic=281848.0