Compact Confidential Transactions for Bitcoin

Denis Lukianov*

Published 9th June, Revised 22th December 2015

Abstract

An enhancement is suggested to make Bitcoin[1] transaction amounts hidden to all but the sender and receiver. In each transaction, the output amounts are encrypted with the public keys of the respective receivers. Only the transaction fee is publicly revealed, to allow miners to prioritise transactions. A homomorphic commitment for each transaction proves that the sum of the transaction inputs matches the sum of its outputs. A short Non-Interactive Zero-Knowledge Proof (NIZKP) for each output also convinces all verifiers that the sum does not overflow. Address construction includes an additional public view key to allow senders to encrypt output values. This approach practically resolves a core privacy issue in Bitcoin, but without overwhelming implementation complexity. The required commitments are an order of magnitude smaller than those proposed for Confidential Transactions[2], and do not depend on ring signatures.

1 Introduction

As Bitcoin[1] continues to generate interest in both academic and business circles, an increasing attention is being given to its pseudonymous properties. The question arises with regards to the levels of anonymity that can be achieved by a cryptocurrency, and the implementation cost of such protection.

1.1 Privacy

Privacy was a low priority in the Bitcoin implementation[3]. A brief look at the source code shows that a node's network address is openly shared with

*BM-2cWtzfLzPdzRcv8CDFTTRSHaSNbFfC3C6V
1SumKArxoEJ1HoGibmj8ygw1DZWYBvjmM
QR code generated with `http://bitcoinqrcode.org/`

other nodes, and at least one public internet service. Some early versions of the software could pay to a specific network address, and connected to a public Internet Relay Chat channel to discover peering nodes.

Many in the cryptocurrency community consider anonymity to be a necessary property of electronic cash. Some users have been willing to risk their coins in inconvenient or fee-based anonymising mixers[4][5][6]. Despite evident need, no cryptocurrency has succeeded in providing strong user anonymity. This is because increased anonymity often implies an implementation cost for the software and usability cost for the user.

1.2 Fungibility

It has long been established that fungibility is a critical aspect assuring the free circulation of cash[7]. Because electronic cash transactions connect specific inputs and specific outputs, the fungibility of a specific output can be questioned. The fungibility of a coin can be influenced by accusations aimed at the source address, or even based on implied the wealth of its owner. At least one developer has attempted to add a coin black list into an operating system distribution[8]. Other attempts at imputing the value of coins based on their source[9] continue to be made[10][11][12][13][14][15]. If coins are made more anonymous, they are also made more fungible, and thus more suitable for commerce.

2 Background

2.1 Mixers

There have been a number of trustless coin mixing algorithms (including CoinJoin, Dash[16] and CoinShuffle[17]) aimed at bringing anonymity to Bitcoin transactions. A big benefit of CoinJoin is that it requires no change to the Bitcoin transaction protocol, and has worked from Bitcoin's inception. Coin mixing does require extra software, and bigger transactions, with only a modest gain in privacy. A significant taint[18] could apply to all coins spent through a mixer, and does not necessarily improve fungibility.

2.2 Related Work

Stealth Addresses[19] hide a transaction's destination address. The sender performs an Elliptic Curve Diffie-Hellman handshake between the receiver's public key (available on the blockchain), and an internally generated nonce, to derive a new key known only to the sender, but detectable by the receiver given the nonce. The public part of the nonce is included in the transaction, and by testing every such transaction output nonce on the blockchain against its own keys, the receiver is able to find the relevant derived keys. CryptoNote[20], and subsequently Monero[21] make use of ring signatures to also hide the transaction's source address.

Some works have shown that it is possible to construct decentralised ledgers which preserve stronger anonymity. Both Zerocoin[22][23] and Zerocash[24] begin by minting anonymised tokens, rather than protecting the base coins. Of known proposals, only Zerocash and CoinWitness[25] take the step of also hiding transaction values. Hiding these values is critical because it ensures that fungibilty of a coin must remain independent of the amount in its owning address. If the value amounts are public, it becomes possible to target identity discovery resources towards wealthy senders.

Zerocoin provably deposits base coins to an accumulator token, and as the order of deposit is not necessarily similar to the order of withdrawal, ownership is obscured. There are down sides to this approach, firstly that Zerocoin transactions are costly in terms of size. There is considerable complexity in new minting and pouring transactions. The full Zerocash paper is over 50 pages long, so achieving a fault-free software implementation of such a complex specification would be resource intensive. It would be challenging to convince those outside the academic sphere of its correctness. The trust placed in the implementation of zkSNARKs and initial trusted set-up are a high cost of these systems. CoinWitness relies even more on novel concepts, which are yet to mature.

Recently, and parallel to the development of this paper, a concept of Confidential Transactions[2] has been introduced, where the transaction value is hidden by Pedersen commitment. These commitments depend on ring signatures for overflow prevention. Some information about the exponent is revealed to keep the proof size manageable. Even so, the storage required remains an order of magnitude larger than for the method proposed in this paper.

2.3 Privacy and Transmission

While securing coins mathematically, few proposals give practical consideration to real world methods of de-anonymising users. If the network node originating a transaction can be traced using time-correlation, or simpler methods, the transaction sender's network identity can be discovered, regardless of how well their coin addresses are mathematically hidden. Often, "the first node to inform you of a transaction is likely to be the source of it"[3]. Tackling the network transmission problem alone, would be a significant practical step toward anonymous transactions, without any changes to the transaction logic.

Anoncoin[26] and others have taken ad-hoc approaches in this direction by interfacing with Tor and I2P. Their approach attempts to hide the physical network address of all nodes, but for transactions this does no better than obfuscate their origin. Transmission through popular routing systems such as Tor is still subject to flow-correlation[27], Sybil and other attacks[28].

This paper does not aim to solve the transmission problem, but to make it less relevant. Zerocash employs complexity to hide all aspects of a transaction. However, address hiding does not necessarily anonymise a user, and privacy given by hiding the value may be of more practical benefit.

2.4 Homomorphic Schemes

It has been has suggested[29] that using homomorphic encryption could effectively hide transaction values. Partial homomorphic encryption schemes such as that of Paillier[30] and others, have been tested thoroughly, and commitment based Non-Interactive Zero Knowledge Proofs (NIZKP) have been peer reviewed for a long[31] time. The schemes have shown us that provably correct calculation can be efficiently perfomed on variables, without knowing their actual value. The additive homomorphic property seems sufficient to verify that the sum of the outputs do not exceed the sum of the inputs in a cryptocurrency transaction. However, as homomorphic schemes are commonly constructed moduli some number, they wrap to zero after exceeding this modulus, so it is also necessary to check transaction outputs for overflow.

The Paillier[30] probabilistic encryption, results in a different ciphertext for every encryption of the same plaintext. An improvement to the system prevents ciphertext expansion[32]. However, despite the improvement, the Elliptic Curve Cryptography is thought to achieve an order of magnitude higher level of security for the same bit length[33]. ECC addition is deterministic rather than probabilistic.

2.5 Compact Proofs

Non-Interactive Zero Knowledge Proofs are arguments about numeric values. Definitive proofs tend to have a large storage requirement. Fortunately, there are several known proofs. Boudot[35] shows how to construct a proof that an encrypted number lies within an interval, without revealing the number itself, for cryptosystems where the group order is known (e.g. Elliptic Curves). Chan, Frankel and Tsiounis[34] Zhengjun[37] show how to construct very compact proofs in groups of unknown order (implied trusted set-up). This is achieved by allowing for an expansion factor between the condition under which the proof is generated, and what it proves. While a narrow interval must be known to construct the proof, the verifier can only be convinced of a much widened interval. The methods have different expansion factors, leading to different efficiencies, which have been compared[36].

3 Construction

3.1 View and Spend Keys

While a single raw public key could be used as the address (and serve as both the spend and view key), this would prevent the user from applying independent security policies to the control and visibility of coins. The separation is necessary to authorise wallet software to report balances, without giving it the power to spend. Dual keys have been useful in the Monero[21] project. In the new scheme, the receiver's public view key is used by the sender to encrypt the transaction output value. The spend key is required, but not sufficient, to spend coins. The previous output value must also be known, so that proofs about the next output value can be constructed. The spend value can of course be cached in wallet software, once it is revealed with the help of the secret view key. The separation of control and visibility should be extended to support multi-signature transactions, where visibility can be granted to a subset of participants.

3.2 Address

If privacy is the priority, it is sufficient to construct a Bitcoin address and distribute it in concatenation with the public view key. The view key does not need to be included in the blockchain. However, this makes payments deniable. That is, a sender could feign payment by switching the public view key and burning the coins. Moreover, a receiver can claim the coins were burned, when they were not.

If the participants value un-deniability over view key un-linkability, the public view key can optionally be included in the spend script. Then sender will be able to prove that the coins are recoverable by the receiver, by presenting only the transaction and destination address. An un-deniable, but less private confidential address can then be constructed:

$$keyHash = Ripemd160(spendScript|Ripemd160(pubViewKey))$$

$$transactionAddress = keyHash|Checksum(keyHash)$$

$$undeniableAddress = base58(pubViewKey|transactionAddress)$$

3.3 Value Commitment

Rather than the value itself, only commitment to an output value is be stored in the transaction output. The value commitment is a point on the Elliptic Curve, which is calculated by a point multiplication of the curve generator by the value:

$$commitment = value * G$$

3.4 Blinding Nonce

The coin value needs no more than 64 bits (*valuebits*), and a reasonably secure point representation can provide more than this (e.g. 256 bits). The remaining bits carry no useful information (effectively sub-satoshis with negligible economic significance), and some are set randomly with a nonce of uniform distribution to blind the value, making it resistant to brute force attacks:

$$fuzz \sim U(0, 2^{fuzzbits})$$

$$fuzzedValue = value * 2^{fuzzbits} + fuzz$$

$$commitment = fuzzedValue * G$$

3.5 Proof of Sum

The probabilistic behaviour is only simulated, so commutative and deterministic properties of Elliptic Curve math are retained. This means an implicit proof of sum is available with no storage cost over that of the commitments. Non-coinbase transactions refer to a previous transaction for their input commitments, and the fee is public. For coinbase transactions, the input is public. For each transaction input, the commitment is made available in the block chain. Checking the following equality verifies the sum magnitude of the outputs matches the sum magnitude of the inputs:

$$\sum_{i=1}^{inputs} vinCommit_i + coinbase * G = \sum_{j=1}^{outputs} commitment_j + fee * G$$

A large output magnitude could cause the calculation to exceed the group order, causing an overflow. This sum proof allows an overflow on the output side of the equation.

3.6 Overflow Free Sum

To prove to all network participants that the sum does not overflow, it is also necessary to construct a proof that each output value is both positive and small enough in magnitude that their sum can not overflow. The upper 8 bits on the curve are reserved to handle overflow up to some maximum number of outputs per transaction, $maxOutputs = 2^{reservedbits} - 1 = 255$.

3.7 Minter Privacy

Coinbase transactions may choose any publicly verifiable deterministic constant to initialise the fuzz (zero results in a round number for satoshi quantity). The Coinbase total can be randomly partitioned between multiple outputs by an algorithm (such as the one included in Appendix A), to improve privacy of the coinbase outputs:

$$cbvalue_{0,\ldots,outputs} = Distribute(outputs, coinbase, valuebits)$$

If coinbase subsidy could be both randomised similar to Luckycoin[41] (and earlier version of Dogecoin[42]), and hidden while proved in a narrow range, this could provide extra initial privacy for the miners. This is considered too expensive to implement. The coinbase is instead constrained to be spent into a minimum of 3 outputs. The constraint ensures that a miner's payee will not be able to determine the exact amounts sent to other payees from the single transaction output.

3.8 Script and Multi-signature Transactions

The coin scripting language is unaffected. If un-deniability is required as part of address construction, the verify opcodes may accept view key as an additional unused parameter. For N-of-M transactions, the view keys can be shared as necessary.

3.9 Sender and Receiver Responsibilities

Sender and receiver must not disclose the view key, amount and fuzz bits used in each transaction. It is up to the sender of a transaction to guarantee its secrecy by generating good randomness for the fuzz bits of each output. Once the details of a transaction are made public, it is likely that they can not be hidden again.

4 Trustless Construction with BCDG

This version is suitable for public proof-of-work blockchains.

4.0.1 Proof of Small Magnitude

The BCDG[36] proof is adapted to the Elliptic Curve, non-interactive setting. Prover and Verifier agree on elliptic curve C of prime order N and generator

G. They also agree on a hash function $Hash$ and security parameter[1] $t = 128$. Prover and Verifier both know a point commitment to x: $F = x * G$. The prover also knows that $x < b$ where b is a public integer. The security parameters imply expansion of $T = 3 * b$ between what is required to construct the proof, and what the Verifier can be convinced of. It is expected that $T < N/(maxOutputs * 3)$, to prevent overflow in the transaction.

Protocol $PKSmall(G, x, F : F = x * G \wedge x \in [-T, 2 * T])$

1. Prover chooses random $u_k \in [0, T]$ and then computes $v_k = u_s - b_k$ for $k \in \{0, ..., t-1\}$.
2. Prover permutes u_k, v_k for each k, so that each pair is unordered.
3. Prover calculates the commitments, $U_k = u_k * G$ and $V_k = v_k * G$.
4. Then she computes $c = Hash(F|U_0|...|U_{t-1}|V_0|...|V_{t-1}) \mod 2^t$.
5. Then for each bit $k \in \{0, ..., t-1\}$ of c:
   If $c_k = 0$ and $u_k \in [0, b]$, Prover sends u_k, and $q_k = 0$.
   If $c_k = 0$ and $u_k \notin [0, b]$, Prover sends u_k, and $q_k = 1$.
   If $c_k = 1$ and $x + u_k \in [0, b]$, Prover sends $y_k = u_k + x$ and $q_k = 0$.
   If $c_k = 1$ and $x + u_k \notin [0, b]$, Prover sends $y_k = v_k + x$ and $q_k = 1$.
6. Finally she sends c to the Verifier.
7. Verifier calculates for each bit $k \in \{0, ..., t-1\}$ of c:
   If $c_k = 0$ and $q_k = 0$, then $U_k = u_k * G$ and $V_k = U_k - b * G$.
   If $c_k = 0$ and $q_k = 1$, then $V_k = u_k * G$ and $U_k = V_k - b * G$.
   If $c_k = 1$ and $q_k = 0$, then $y_k \in [0, b]$, $U_k = y_k * G - E$ and $V_k = U_k - b * G$.
   If $c_k = 1$ and $q_k = 1$, then $y_k \in [0, b]$, $V_k = y_k * G - E$ and $U_k = V_k - b * G$.
8. The verifier then reconstructs the challenge from the commitments and checks that the challenge is c.

4.0.2 Implementation Efficiency

Instead of sending u_k (256 bits) directly, the Prover can generate and send $s_k \in [0, 2^t]$ (128 bits), then let the verifier calculate the same $u_k = Hash(s_k) \mod b$. Further, u_1, u_2 can be calculated by hashing a common secret $z_1 \in [0, 2^t]$ with $u_k = Hash(z_1|0) \mod b$ and $u_2 = Hash(z_1|1) \mod b$. Then if the protocol requires a reveal of both u_1 and u_2, then s_1 and s_2 can be calculated from z_1. More generally, a binary tree of hashes can be used.

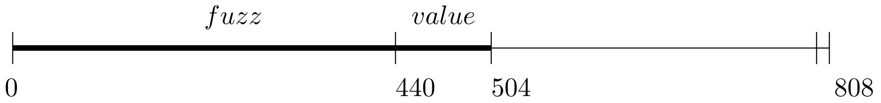
[1] The Schnorr soundness security parameter t determines the probability of a cheating prover succeeding as 2^{-t}.

5 Trusted Construction with CFT

This version is more compact, but only suitable for private blockchains.

5.0.1 Large Curve

A larger curve is required to accommodate squaring for the CFT proof. The bit representation can be visualised from low bits on the left to high bits on the right:



5.0.2 Substitute Commitments

A zero knowledge proof based on CFT[36] is constructed. The sender, instead of committing to $fuzzedValue$, commits to a square number close to $fuzzedValue$. An expensive search[40] to find three squares of the exact large integer is avoided, if every output is represented as a square:

$$x = \lfloor \sqrt{fuzzedValue} \rfloor$$
$$\Delta = fuzzedValue - x^2$$

The sum of Δ_j for the outputs can be revealed at the transaction level, as part of the fee, and masked by randomness in the fee. Outputs can only be squares in the scheme, but this is not a limitation, because $\sqrt{fuzzedValue}$ is too insignificant to leak into the *value* bits.

x has half the bits of $fuzzedValue$, with the lower $fuzzbits/2$, or 220 bits, still random on average[2]. The sender (Prover) must prove to the network participants (Verifier), in zero knowledge, that the commitment to x^2 really is the square of the commitment to x, and that x is sufficiently small that the sum of x_j^2 and Δ_j does not overflow.

The following commitments to x are made:

$$E = x * G$$

$$F = x * E$$

The fuzz bits can be thought of as blinding sub-satoshis. Inputs are outputs to the next transaction, so there is a fixed bound on the possible value within the system, which is verified to be unchanged by the transaction:

$$\sum_{i=1}^{inputs} vinCommit_i + coinbase * G = (\sum_{j=1}^{outputs} F_j) + (fee + \sum_{j=1}^{outputs} \Delta_j) * G$$

[2] In the worst case, this is 195 bits, because the root is rounded down.

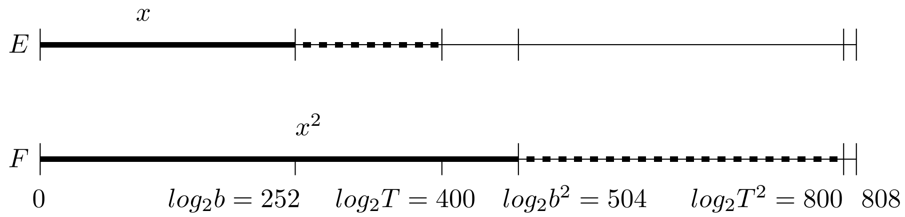
5.0.3 Proof of Square and Small Magnitude

The CFT[36] proof is adapted to the Elliptic Curve setting. Prover and Verifier both know a third party who is trusted to generate random numbers within a limited range. Prover and Verifier agree on elliptic curve C of prime order N and generator G. They also agree on a hash function $Hash$ and security parameters[3][4] $t = 128$ and $l = 20$. Prover and Verifier both know point commitments to x: $E = x * G$, $F = x * E$. The prover also knows that $x < b$ where b is a public integer. The security parameters imply expansion of $T = 2^{t+l}b - 1$ between what is required to construct the proof, and what the Verifier can be convinced of. It is expected that $T < \sqrt{N}/maxOutputs$, to prevent overflow in the transaction.

Protocol $PKSqSmall(G, x, E : E = x * G \wedge F : F = x * E \wedge x \in [-T, T])$

1. Prover obtains a random $r \in [0, T]$ and $U = r * G$ from the trusted third party, with a signature[5] S on U .
2. Prover computes $V = r * E$.
3. Then she computes $c = Hash(E|F|U|V) \mod 2^t$.
4. Finally she computes $m = r + c * x$.
5. If $m > c * b$, $m < T$, she sends[6] (m, c, S) to Verifier, otherwise she starts again the protocol.
6. Verifier checks that U was signed by the trusted third party by checking the signature S.
7. Verifier calculates
   $P = m * G - c * E$ and $Q = m * E - c * F$.
8. Verifier checks that $c * b < m < T$ and $c = Hash(E|F|P|Q)$, which convinces Verifier that $P = U$, $Q = V$, $x \in [-T, T]$ and therefore $F = x * E = x^2 * G$.

The bit representations of x and x^2 do not use up the whole curve, to allow the proof to be constructed:



[3] The Schnorr soundness security parameter t determines the probability of a cheating prover succeeding as 2^{-t}.

[4] The Zero Knowledge security parameter l determines the effectiveness of statistical attack on m as related to 2^{-l}. On average, many multiples of 2^l proofs must be observed to contemplate an attack.

[5] The signature can be on a smaller curve

[6] The original protocol also sent points, but this is not necessary as they can be computed.

6 Transaction Output Format

In addition to signature and script, and instead of the publicly visible value, the enhanced transaction output contains several elements to prove the validity of the transaction to the network:

- $voutCommit_j$: public commitments to the output value, including the fuzz bits, represented as points:

$$voutCommit_j = F_j$$

- $voutProof_j$: a compact NIZKP proof of unsigned smallness on the committed value (see proofs).
- $voutCrypt_j$: a one-time encryption of the value as performed by the sender by Elliptic Curve Diffie-Hellman exchange using x_j as the private key for the source transaction, and the receiver's public view key:

$$commonSecret_j = x_j^2 * voutViewPubKey_j$$

$$voutCrypt_j = Hash(commonSecret_j) \oplus x_j$$

  The receiver can later obtain the same shared secret from the transaction by using her private view key:

$$commonSecret_j = voutViewPrivKey_j * F_j$$

6.1 Transaction Size

For each output, the value hiding enhancement adds about 352 bytes, and not sending the plain value removes 8, for a net change of 344 bytes:

- E and F (102 bytes each)
- m (50 bytes)
- c (16 bytes)
- $voutCrypt$ (32 bytes)

The extended fee claims 64 bytes at the transaction level, instead of 8. Only one commitment F (102 bytes) needs to be kept in each unspent transaction output. At the user level, value hiding can encourage fewer transaction outputs to maintain the same level of privacy.

Since the introduction of multi-signature addresses, the average Bitcoin transaction size has risen to about 600 bytes. For a typical two-input, two-output transaction, the hiding overhead is then about 704 bytes (+117%).

These numbers are a guide, and are not absolute. There is an adjustable tradeoff between the bit-security preventing forgeability, as defined by security parameter t; the statistical security defined by security parameter l; and the bit-security of value hiding, due to the security parameter $fuzzbits$. There are also adjustable tradeoffs between the security bit-levels, commitment size, curve order and $maxOutputs$. Special cases such as single-input single-output transactions do not require expensive proofs.

Note that ultra-small value transactions ("dust") cannot be rejected, because their value is not known. To prove that each output is big enough to be economically important, would further increase the transaction size and verification time. Instead, a positive non-zero transaction fee can be mandated by the miners to protect the ledger from abuse (note that as of version 0.1, Bitcoin encourages non-zero fees). Beyond the scope of this paper, dust can be also be mitigated by changing transaction incentives to reduce the size of the unspent output set.

6.2 Computational Load

The value verification requires, for every transaction input and output, three ECC additions to verify the sum. For every output, the verification of the proof requires three ECC add-multiply steps and three ECC additions.

Constructing a transaction requires three ECC multiplications for every output. Constructing an output proof requires four (or probabilistically slightly more) ECC multiplications and two ECC addition. Transaction construction is a much less common task than transaction verification.

The performance impact of decrypting the encrypted value is small, and this need only performed for transactions in the local wallet.

Fortunately, despite the large order of curve, the scalars used in multiplication are small, and this has a positive impact on performance. To avoid time consuming multiplication by a negative scalar, a point can be inverted over the y axis.

6.3 Comparison to CT

CCT was tested using OpenSSL on Q9550 (and normalised by factor 1.82 for i7-4770R) and compared to the published[2] CT result. CT uses libsecp256k1, which is a much faster curve-specific library. Thus a lot of scope for CCT optimisation remains.

Property	**CT**	**CCT**	**Improvement**
value bits hidden	32	64	100%
blockchain space, kB	2.55	0.35	728%
verifications, per sec	1300	600	-54%

7 Implications

7.1 Features

Audit is still possible through equivalence proofs, or trivially by sharing a view key. Transaction value statistics are hidden, though the total number of coins created is known, because the coinbase values are public. The enhancement does not provide support for the creation of Coloured Coins[44], without revealing values. In addition to standing on its own, the enhancement can be implemented as a sidechain[45] or integrated into the Bitcoin protocol as a hard fork with a new transaction version. Spent transaction pruning[46] is still possible with the enhancement.

Of course, the enhancement does not preclude a user from using additional address hiding protocols such as mixing, though the linkability of both the view key and the spend key must be considered. A hidden satoshi will mix as well as a hidden coin, so this enhancement further improves CoinJoin. Hidden values deprecate privacy preserving strategies such as merge avoidance[43]. This scheme, of course, does not depend on a tiered network architecture, but can be integrated into Dash[16]. A user need not mix coins of related denominations, she can simply spend to multiple addresses.

This method does not hide all aspects of a transaction like like ZeroCash, but by hiding the amount, it does hide the most important aspect, while avoiding the complexity of zkSNARKS and additional initialisation functions.

If almost all inputs or outputs in any transaction are revealed, this will also reveal the remaining input (or output). This effect can propagate to related transactions if they are themselves almost fully revealed.

7.2 Social

The enhancement makes Bitcoin more like cash. Users of cash tend not to discover the value of other user's transactions. Without this enhancement, Bitcoin nodes discover the value of each unrelated transaction on the global network.

With cash transactions, if a party chooses to publicly disclose a transaction amount, their claim would initially stand unconfirmed. They would need additional evidence, such as a confirmation from the counterparty (or intermediary) to try to back up the claim. With Bitcoin, such disclosure is immediately and forever provable on the blockchain. This permanent and undeniable record should discourage the use of this technology for nefarious purposes.

Properly kept, crypto-currencies such as Bitcoin are the most difficult asset class to take from an owner without consent. This process reduces to probabilistic rubber hose cryptanalysis in extremis, which may only be feasible on a small scale. Unfortunately, not all owners can be assumed to make sufficient effort to protect their coins, and the public visibility of Bitcoin values make the owners a target[47]. This enhancement to Bitcoin hides value to discourage the selection and prioritisation of targets for the application of cryptanalysis.

8 Conclusion

It has been shown to be not only possible, but very practical, to enhance a crypto-currency like Bitcoin to hide transaction values from public view, while maintaining the integrity of each transaction, with only a small increase in computational and storage requirements over the non-hiding protocols. Zero-knowledge proofs are performed on the elliptic curve by the sender of every transaction, these convince all honest nodes that the sum of the transaction outputs equals the sum of the transaction inputs, and that overflow did not occur. Elliptic point commitments include a random fuzz component to deflect brute force attacks, and this component is uniformly rebalanced in the outputs of each transaction. This design is significantly simpler to implement than other methods proposed thus far. Some implications of the technology are discussed.

9 Acknowledgements

Thanks to Jochen Hoenicke for suggesting cryptanalytic approaches, suggesting that on a large curve, the single square output approach performs better than a sum of two squares and fixing a bunch of other issues in the paper. Thanks to Andrew Poelstra for breaking an initially over-optimistic proof of smallness. Thanks to Jonathan Bootle for breaking the proof by committing to a multiplicative inverse modulo the known group order. Thanks to IC3 for extending the multiplicative inverse break to include odd challenges. Thanks to Gregory Maxwell for a brief review.

References

[1] Satoshi Nakamoto
Bitcoin: A Peer-to-Peer Electronic Cash System
`https://bitcoin.org/bitcoin.pdf`
`https://github.com/bitcoin/bitcoin`

[2] Gregory Maxwell, Adam Back
Blockstream Elements: Confidential Transactions
`https://people.xiph.org/~greg/confidential_values.txt`
`https://bitcointalk.org/index.php?topic=1085273.0`
`https://blockstream.com/developers/`

[3] Fergal Reid and Martin Harrigan
An Analysis of Anonymity in the Bitcoin System
`http://anonymity-in-bitcoin.blogspot.co.uk/2011/07/bitcoin-is-not-anonymous.html`

[4] invisibel
TiPS - built-in coin mixing/anonymity service
`http://fedoracoin.net/`
`https://bitcointalk.org/index.php?topic=436467.40`

[5] CoinCidental
Bitcoin Blender, anonymous bitcoin mixer
`https://bitcointalk.org/index.php?topic=436467.40`

[6] Coinblender
Bitcoin Blender, anonymous bitcoin mixer
`https://coinblender.net`

[7] Kenneth Reid
Banknotes and Their Vindication in Eighteenth-Century Scotland, 2013
`https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2260952` David Fox and Wolfgang Ernst (eds), Money in the Western Legal Tradition (Oxford University Press, 2015) ISBN 0198704747

[8] Luke-Jr
Luke-Jr's public apology for poor Gentoo packaging default
`http://www.reddit.com/r/Bitcoin/comments/2iuf4s/lukejrs_public_apology_for_poor_gentoo_packaging/`

[9] Ale Janda
Bitcoin block explorer with address grouping and wallet labeling
`https://www.walletexplorer.com/`

[10] Forbes / Coin Validation
Sanitizing Bitcoin: This Company Wants To Track "Clean" Bitcoin Accounts
`http://www.forbes.com/sites/kashmirhill/2013/11/13/sanitizing-bitcoin-coin-validation/`

[11] Michael Grnager, Jan Mller
`https://chainalysis.com/`

[12] Jimmy Thommes
Evolution Marketplace Collapse Violates Bitcoin Fungibility
`http://news.diginomics.com/evolution-marketplace-collapse-violates-bitcoin-fungibility/`

[13] Carlo Caraluzzo
Coinbase Is Tracking How Users Spend Their Bitcoins
`https://cointelegraph.com/news/113207/coinbase-is-tracking-how-users-spend-their-bitcoins`

[14] Emily Spaven
Accenture: UK Government Should Regulate Bitcoin Wallets
http://www.coindesk.com/accenture-uk-government-should-regulate-bitcoin-wallets/

[15] Dr. James Smith
https://www.elliptic.co

[16] Evan Duffield, Damiel Diaz
Dash: A Privacy Centric Crypto Currency
https://www.dashpay.io/wp-content/uploads/2015/04/Dash-WhitepaperV1.pdf

[17] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate
CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin
https://crypsys.mmci.uni-saarland.de/projects/CoinShuffle/coinshuffle.pdf
https://bitcointalk.org/index.php?topic=567625.0

[18] Johannes Schweifer
The taint and the Bitcoin
https://www.bitcoinsuisse.ch/en/the-taint-and-the-bitcoin/

[19] ByteCoin, Peter Todd, et al.
[Bitcoin-development] Stealth Addresses
http://sourceforge.net/p/bitcoin/mailman/message/31813471/
https://bitcointalk.org/index.php?topic=5965.0
http://sx.dyne.org/stealth.html

[20] Nicolas van Saberhagen
CryptoNote v 2.0, October 17, 2013
https://cryptonote.org/whitepaper.pdf

[21] Monero developers
Monero
https://getmonero.org/home
https://bitcointalk.org/index.php?topic=583449.0

[22] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin.
Zerocoin: Anonymous distributed e-cash from bitcoin
In Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP 13, pages 397411, 2013.
http://zerocoin.org/media/pdf/ZerocoinOakland.pdf

[23] George Danezis, Cdric Fournet, Markulf Kohlweiss, Bryan Parno
Pinocchio Coin: Building Zerocoin from a Succinct Pairing-based Proof System
http://research.microsoft.com/en-us/um/people/fournet/papers/pinocchio-coin.pdf

[24] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza
Zerocash: Decentralized Anonymous Payments from Bitcoin, IEEE SSP 2014
http://zerocasNewh-project.org/media/pdf/zerocash-oakland2014.pdf

[25] Gregory Maxwell
Really Really ultimate blockchain compression: CoinWitness
https://bitcointalk.org/index.php?topic=277389.0

[26] Anoncoin
Anoncoin
http://www.anoncoin.net/

[27] Sambuddho Chakravarty, Marco V Barbera, Georgios Portokalidis, Michalis Polychronakis, Angelos D Keromytis

On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records
Passive and Active Measurement 2014/01/01, p247-257
`https://mice.cs.columbia.edu/getTechreport.php?techreportID=1545&format=pdf`

[28] Alex Biryukov, Ivan Pustogarov
Bitcoin over Tor isnt a good idea, 8 January 2015
`http://arxiv.org/pdf/1410.6079v2.pdf`

[29] Adam Back
bitcoins with homomorphic value (validatable but encrypted)
`https://bitcointalk.org/index.php?topic=305791`

[30] Pascal Paillier
Public-Key Cryptosystems Based on Composite Degree Residuosity Classes
Proc. EUROCRYPT99, Springer-Verlag, LNCS 1592 (1999).
`https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.8294`

[31] Shafi Goldwasser, Silvio Micali, and Charles Rackoff
The Knowledge Complexity of Interactive Proof Systems
SIAM Journal on Computing (1989) 18(1), 186208.
`https://groups.csail.mit.edu/cis/pubs/shafi/1985-stoc.pdf`

[32] Ivan Damg ard, Mads Jurik and Jesper Buus Nielsen
A Generalization of Pailliers Public-Key System with Applications to Electronic Voting
`https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.3383`

[33] NIST Recommendations 2012
`http://www.keylength.com/en/4/`
`http://csrc.nist.gov/groups/ST/toolkit/key_management.html`

[34] Chan, A., Frankel, Y., Tsiounis, Y
Easy Come - Easy Go Divisible Cash. Proceedings of EUROCRYPT98, LNCS 1403 (1998) 561575
`https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.8370`

[35] Brickell, E., Chaum, D., Damgrd, I., Van de Graaf, J.:
Gradual and Verifiable Release of a Secret. Proceedings of CRYPTO87, LNCS 293 (1988) 15616
`https://www.researchgate.net/publication/226900963_Gradual_and_Verifiable_Release_of_a_Secret_Extended_Abstract`

[36] Fabrice Boudot
Efficient Proofs that a Committed Number Lies in an Interval
`https://www.iacr.org/archive/eurocrypt2000/1807/18070437-new.pdf`

[37] Zhengjun Cao
An Efficient Range-Bounded Commitment Scheme
`https://eprint.iacr.org/2007/376.pdf`

[38] Fabrice Boudot and Jacques Traor
Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery, ICICS Proceedings November 1999, page 88

[39] Warren D. Smith
"ZK-proof for a sum of squares", Cryptography meets voting, 2005
`https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.3282`

[40] Michael O. Rabin & Jeffrey O. Shallit
Randomized Algorithms in Number Theory, Communications Pure & Applied Mathematics 39 (1986) 239-256

[41] LuckyC, Titan
Luckycoin Cryptocurrency
https://luckycoin.cc/
https://bitcointalk.org/index.php?topic=568287

[42] Jackson Palmer, Shibetoshi Nakamoto
Dogecoin
http://dogecoin.com/
https://en.wikipedia.org/wiki/Dogecoin

[43] Mike Hearn
Merge avoidance: Privacy-enhancing techniques in the Bitcoin protocol, December 2013
https://medium.com/@octskyward/merge-avoidance-7f95a386692f
http://www.coindesk.com/merge-avoidance-privacy-bitcoin/

[44] http://coloredcoins.org/

[45] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timn, and Pieter Wuille
Enabling Blockchain Innovations with Pegged Sidechains, 2014-10-22
https://blockstream.com/sidechains.pdf

[46] B.F.Franca
Privacy and pruning in the Mini-blockchain
http://cryptonite.info/files/Anonymity_account_tree.pdf

[47] Ben Hunt
The Effete Rebellion of Bitcoin
http://www.zerohedge.com/news/2015-02-19/guest-post-bitcoin-effete-act-rebellion
http://www.salientpartners.com/epsilontheory/post/2015/02/17/the-effete-rebellion-of-bitcoin

A The Distribute algorithm

Algorithm 1 To partition a *total* into *j* random parts while preserving the sum

```
procedure DISTRIBUTE($j, total, bits$)
    $Assert(0 < j < total)$                                   ▷ input validation
    $Assert(j * 2^{bits-2} < total < j * 2^{bits})$
    $mask = 2^{bits} - 1$
    for $x = 1$ to $j$ do
        $\mathrm{ret}_j \leftarrow 0$                          ▷ initialise return values
    end for
    while $total > 1$ do                          ▷ work until $total$ is fully disbursed
        $\mathrm{fences}_0 \leftarrow 0$
        $\mathrm{fences}_{j+1} \leftarrow total$
        for $x = 1$ to $j$ do                            ▷ random without replacement
            $\mathrm{fences}_x \leftarrow Random(U(1, total) \cap \mathrm{fences}_{0,...,x-1})$
        end for
        $\mathrm{fences}_{0,...,j+1} \leftarrow Sort(\mathrm{fences}_{0,...,j+1})$
        for $x = 0$ to $j$ do                     ▷ partition the total into $delta$s
            $\mathrm{delta} \leftarrow \mathrm{fences}_{x+1}$ - $\mathrm{fences}_x$
            $\mathrm{ret}_x \leftarrow \mathrm{ret}_x + \mathrm{delta}$
            $total \leftarrow total$ - delta
            while $\mathrm{ret}_x > mask$ do                    ▷ carry $delta$ overflow
                $\mathrm{ret}_x \leftarrow \mathrm{ret}_x$ - $mask$
                $total \leftarrow total + mask$
            end while
        end for
    end while
    return $\mathrm{ret}_{0,...,j}$                   ▷ implicitly, $total \equiv \sum ret_{0,...,j}$
end procedure
```