Confidential Assets

Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille

Blockstream
{apoelstra, adam, mark, gmaxwell, pwuille}@blockstream.com

Abstract. Bitcoin is an online distributed ledger in which coins are distributed according to the *unspent transaction output (UTXO)* set, and transactions describe changes to this set. Every UTXO has associated to it an amount and signature verification key, representing the quantity that can be spent and the entity authorized to do so, respectively.
Because the ledger is distributed and publicly verifiable, every UTXO (and the history of all changes) is publicly available and may be used for analysis of all users' payment history. Although this history is not directly linked to users in any way, it exposes enough structure that even small amounts of personally identifiable information may completely break users' privacy. Further, the ability to trace coin history creates a market for "clean" coins, harming the fungibility of the underlying asset.
In this paper we describe a scheme, *confidential transactions*, which blinds the amounts of all UTXOs, while preserving public verifiability that no transaction creates or destroys coins. This removes a significant amount of information from the transaction graph, improving privacy and fungibility without a trusted setup or exotic cryptographic assumptions.
We further extend this to *confidential assets*, a scheme in which a single blockchain-based ledger may track multiple asset types. We extend confidential transactions to blind not only output amounts, but also their asset type, improving the privacy and fungibility of all assets.

1 Introduction

Deployed in 2009, Bitcoin [16] is an online currency with no trusted issuer or transaction processor, which works by means of a publicly verifiable distributed ledger called a *blockchain*. The blockchain contains every transaction since its inception, resulting in a final state, the *unspent transaction output set (UTXO set)*, which describes the amounts and owners of all coins.

Each UTXO contains an amount and a verification key; transactions destroy UTXOs and create new ones of equal or lesser total amount, and must be signed with the keys associated to each destroyed UTXO. This model allows all users to verify transaction correctness without trusting any payment processor to be honest or reliable. However, this model has a serious cost to user privacy, since every transaction is preserved forever, exposing significant amounts of information directly and indirectly [10].

One suggestion to obscure transaction structure is CoinJoin [13], which allows users to interactively combine transactions, obscuring which inputs map to which outputs. However, because transaction amounts are exposed, it is difficult to use CoinJoin in such a way that these mappings cannot be recovered, at least in a statistical sense [20]. In particular, unless all output amounts are the same, they are distinguishable and may be grouped.

We propose a partial solution to the exposure of transaction data, which blinds the amounts of all outputs, while preserving public verifiability of the fact that the total output amount is equal to the total input amount. This solution, termed *confidential transactions*, has been described informally by Maxwell [14] and deployed on the Elements Alpha sidechain [2] for over a year. In brief, each explicit UTXO amount is replaced by a homomorphic commitment to the amount. Since these amounts are homomorphic over a finite ring rather than the set of integers, we also attach a rangeproof to each output to prevent attacks related to overflow.

First, we formalize and improve confidential transactions, describing a space optimization of the underlying ring signature used in Elements Alpha. Then we extend confidential transactions to a new scheme, *confidential assets*, which further supports multiple asset types within single transactions. We retain public verifiability that no assets are created or destroyed, while hiding both the output amount(s) and the output asset type(s).

Related Work. Multi-asset blockchains were described in 2013 in Friedenbach and Timón's Freimarkets [8], though the supported assets were not confidential; that is, the amounts and asset tags of all inputs and outputs of transactions are publicly visible.

Support for asset issuance on top of Bitcoin has been proposed by means of *colored coins* [12], a scheme in which individual coins are marked in such a way that they are identifiable as representing distinct asset types. In effect, it works by exploiting Bitcoin's imperfect fungibility.

Ethereum [22] directly supports asset issuance using its smart contracting language, and has a standard means to do so which ensures interoperability with supporting software [18]. Like the above schemes, no attempt is made to obfuscating either the asset types or their amounts.

ZCash [21] is a recently announced cryptocurrency project which supports blinding of amounts, as well as any other identifying information about transaction inputs and outputs. It does not support multiple assets, though its use of zk-SNARKs [3], which are general-purpose zero-knowledge arguments, mean that asset support would not be a difficult extension.

However, ZCash's privacy comes at a significant cost: the underlying SNARKs use a trusted setup, meaning it is initialized by multiple parties who are able to collude to silently inflate the currency; it relies on novel cryptographic assumptions; its zero-knowledge proofs are very slow to compute. To contrast, the scheme described in this paper relies only on elliptic curve discrete logarithm (ECDL) being hard and the random oracle model, and all computations involve few and standard elliptic curve operations (e.g. no pairings).

Acknowledgements. We thank Ben Gorlick for his input on the practical requirements of a confidential assets-based system, and his technical review, and feedback on the systems design.

2 Preliminaries

Definition 1. *We define a* Bitcoin transaction *as the following data:*

- *A list of* outputs*, containing a verification key and an amount.*
- *A list of* inputs*, which are unambiguous references to the outputs of other transactions. These also have signatures using the verification keys of their respective outputs.*
- *A* fee*, which is computed as the total input amount minus the total output amount, and is captured by the network.*

(To bootstrap the system, we also need *coinbase transactions*, which have outputs but no inputs; for the purpose of this paper they can be considered as transactions with negative fee.)

In Bitcoin, all amounts are explicit, and for a (non-coinbase) transaction to be valid, it must have a non-negative fee as well as valid signatures of the transaction with all inputs' verification keys.

We will replace these explicit amounts with homomorphic commitments, for which we need the following definitions.

Definition 2. *Given a message space $\mathcal{M} = \mathbb{Z}_q$, commitment space $\mathcal{C} \simeq \mathcal{M}$ and public parameters space $\mathcal{PP}$, we define a* homomorphic commitment scheme *as a triple of algorithms:*

- Setup$: \cdot \to \mathcal{PP}$
- Commit$: \mathcal{PP} \times \mathcal{M} \times \mathcal{M} \to \mathcal{C}$,
- Open$: \mathcal{PP} \times \mathcal{M} \times \mathcal{M} \times \mathcal{C} \to \{\text{true}, \text{false}\}$

satisfying, for $\text{pp} \leftarrow \text{Setup}$,

- *for all* $(m, r) \in \mathcal{M} \times \mathcal{M}$, $\text{Open}(\text{pp}, m, r, \text{Commit}(\text{pp}, m, r))$ *accepts; and*
- *if*

$$\text{Open}(\text{pp}, m_1, r_1, C_1) \text{ and } \text{Open}(\text{pp}, m_2, r_2, C_2)$$

both accept, then

$$\text{Open}(\text{pp}, m_1 + m_2, r_1 + r_2, C_1 + C_2)$$

also accepts.

We will often leave pp *implicit and not mention it as an input to* Commit *or* Open*. Unless otherwise specified, all theorems are understood to hold for all* $\text{pp} \in \mathcal{PP}$*.*

We further require our commitments be binding and hiding, by which we mean the following.

Definition 3. *A commitment is perfectly binding if for all $m \neq m' \in \mathcal{M}$, all $r, r' \in \mathcal{M}$,* $\text{Open}(m, r, \text{Commit}(m', r'))$ *rejects.*

It is computationally binding if for all p.p.t. adversaries $\mathcal{A}$, the probability of $\mathcal{A}$ producing (m', r') with $m' \neq m$ such that $\text{Open}(m, r, \text{Commit}(m', r'))$ *accepts is negligible.*

Definition 4. *A commitment scheme is (perfectly, statistically, computationally) hiding if given* pp *and $m_1 \neq m_2$, the distributions*

$$U_1 = \{C : C \leftarrow \text{Commit}(\text{pp}, m_1, r), r \xleftarrow{\$} \mathcal{M}\}$$

$$U_2 = \{C : C \leftarrow \text{Commit}(\text{pp}, m_2, r), r \xleftarrow{\$} \mathcal{M}\}$$

are (equal, statistically indistinguishable, computationally indistinguishable).

For the purposes of this paper, we will use Pedersen commitments, which are computationally binding, perfectly hiding homomorphic commitments [17]. They are defined as follows.

Definition 5. *The* Pedersen commitment scheme *is the following triple of algorithms. We take $\mathcal{M} = \mathbb{Z}_q$ and $\mathcal{C}$ to be an isomorphic elliptic curve group; further $\mathcal{H}$ is a point-valued hash function modeled as a random oracle.*

- Setup *takes a cyclic group with distinguished generator $(\mathcal{G}, G)$ as well as auxiliary input α. It computes $H = \mathcal{H}(\alpha)$ and outputs* $\text{pp} = \{\mathcal{G}, G, H\}$.
- $\text{Commit}(m, r)$ *outputs $mH + rG$.*
- $\text{Open}(m, r, C)$ *accepts iff $C = mH + rG$.*

(The original Pedersen scheme uses uniformly random generators G, H, rather than taking H as the output of a hash function. In the random oracle model, these are equivalent.)

In order to commit to transaction amounts, which are integers, we will need to represent them as elements of $\mathcal{M} = \mathbb{Z}_q$, which will complicate matters since every multiple of q will be indistinguishable from zero. To avoid problems, we will need one more primitive.

Definition 6. *Given a homomorphic commitment scheme as above, and $0 \leq A \leq B \leq q$, we define a rangeproof of the range $[A, B]$ as a pair of randomized algorithms*

- $\text{Prove}_{[A,B]}$*: $\mathcal{PP} \times \mathcal{M} \to \mathcal{C} \times \mathcal{M} \times \mathcal{S}$ takes a value and generates a commitment to that value with opening information and an associated rangeproof.*
- $\text{Verify}_{[A,B]}$*: $\mathcal{PP} \times \mathcal{C} \times \mathcal{S} \to \{\text{true}, \text{false}\}$ takes a commitment and rangeproof and either accepts or rejects it.*

where $\mathcal{S}$ represents the space of possible rangeproofs. We require that for all $v \in [A, B]$, $(C, r, \pi) \leftarrow \text{Prove}_{[A,B]}(v)$ that both

$$\text{Verify}_{[A,B]}(C, \pi) \text{ and } \text{Open}(v, r, C)$$

accept.

We require the following security properties of rangeproofs:

Definition 7. *(Proving.) Let $0 \leq A \leq B \leq q$. Then a rangeproof scheme is* proving an amount in the range $[A, B]$ *if for any p.p.t. algorithm $\mathcal{A}$ that outputs $(C, \pi) \in \mathcal{C} \times \mathcal{S}$ such that $\text{Verify}(C, \pi)$ accepts, a simulator $\mathcal{B}$ exists which given oracle access to $\mathcal{A}$ can produce (v, r) such that $v \in [A, B]$ and $\text{Open}(v, r, C)$ accepts.*

We observe that since the commitment scheme is binding, an opening to an amount in $[A, B]$ precludes an opening to any amount outside of $[A, B]$.

In light of Definition 7, given a commitment C with valid rangeproof π, we can talk about "the opening information (v, r) of C" unambiguously in simulation-based proofs, even without knowledge of it (since this knowledge can in principle be obtained by the simulator). In particular, any security proof which requires an adversary to produce opening information of commitments will continue to hold if the opening information is replaced by rangeproofs.

Definition 8. *(Statistical zero-knowledge.) Given $\text{pp} \in \mathcal{PP}$ and two values $v_1, v_2 \in [A, B]$, the following distributions are identical:*

$$\{(C, \pi) : (C, \cdot, \pi) \leftarrow \text{Prove}\,(\text{pp}, v_1)\}$$

$$\{(C, \pi) : (C, \cdot, \pi) \leftarrow \text{Prove}\,(\text{pp}, v_2)\}$$

3 Confidential transactions

3.1 Rangeproofs

We begin by describing an efficient rangeproof for Pedersen commitments over the interval $[0, m^n - 1]$, which has total size proportional to $1 + nm$, using a variant of a folklore bit-decomposition based rangeproof, in which numbers are expressed in base m and each digit is proven to lie in $[0, m - 1]$ using a ring signature.

We use a variant of Borromean Ring Signatures [15], which itself is a variant of the Abe-Ohkubo-Suzuki ring signature [1], tweaked to exploit the fact that many small rings of related keys are used.

Unlike some other rangeproofs in the literature [6], ours does not require a trusted setup [1]. In fact, the only cryptographic assumption it relies on is the

[1] While our rangeproof does require setup, the only generated parameters are uniformly random curvepoints, which can be generated with no possibility of trapdoor information, e.g. by the algorithm by Fouque and Tibouchi [7]

hardness of discrete logarithm in the random oracle model. Nor is it interactive, as is the scheme described in [5]. Despite these improvements, our scheme still produces smaller proofs than these papers for the ranges (30-80 bits) that we are interested in.

Schoenmakers [19] described a simple rangeproof of base-b digits using the conjuction of zero-knowledge OR proofs of each digit. Our work is based on this rangeproof with the following changes: our OR proofs are based on Borromean Ring Signatures, which allow sharing a random challenge across every digit's proof, and we remove one scalar from each proof by a novel trick in which we may change the commitment to each digit (without changing the digit itself) while we produce the proof.

Definition 9. *(Back-Maxwell Rangeproof) Consider a Pedersen commitment scheme with generators G, H, and let $\mathcal{H} : \mathcal{C} \to \mathcal{M}$ be a random oracle hash.*

- Verify $\left(C, \pi = \left\{e_0, \left(C^0, s_1^0, s_2^0, \ldots, s_{m-1}^0\right), \ldots \left(C^{n-1}, s_1^{n-1}, s_2^{n-1}, \ldots, s_{m-1}^{n-1}\right)\right\}\right)$ *works as follows:*
  1. *For each $i \in \{0, \ldots, n-1\}$,*
     (a) *Define $e_0^i = e_0$ for consistency of the following equations.*
     (b) *For each $j \in \{1, \ldots, m-1\}$, compute*

$$e_j^i \leftarrow \mathcal{H}\left(s_j^i G - e_{j-1}^i \left[C^i - j m^i H\right]\right) \tag{1}$$

     (c) *Compute $R^i \leftarrow e_{m-1}^i C^i$.*
  2. *Compute $\hat{e}_0 \leftarrow \mathcal{H}(R^0 \| \cdots \| R^{n-1})$.*
  3. *Accept iff:*
     - *$\hat{e}_0 = e_0$; and*
     - *$C = \sum_i C^i$.*
- Prove(v, r). *Proving works as follows.*
  1. *Write v in base m as $v^0 + v^1 m + \cdots + v^{n-1} m^{n-1}$. (Note that superscripts on m are exponents while superscripts on v are just superscripts.)*
  2. *For each $i \in \{0, \ldots, n-1\}$,*
     (a) *If $v^i = 0$, choose $k_0^i \xleftarrow{\$} \mathbb{Z}_q$ and set $R^i \leftarrow k_0^i G$.*
     (b) *Otherwise,*
        i. *Choose r^i uniformly randomly and compute $C^i \leftarrow \text{Commit}(m^i v^i, r^i)$.*
        ii. *Choose $k^i \xleftarrow{\$} \mathbb{Z}_q$ and compute $e_{v^i}^i \leftarrow \mathcal{H}(k^i G)$.*
        iii. *For each $j \in \{v^i + 1, \ldots, m-1\}$, choose $s_j^i \xleftarrow{\$} \mathbb{Z}_q$, and compute e_j^i directly from equation (1). (If $v^i = m-1$, this step is a no-op.)*
        iv. *Compute $R^i \leftarrow e_{m-1}^i C^i$.*
  3. *Set $e_0 \leftarrow \mathcal{H}(R^0 \| \cdots \| R^{n-1})$.*
  4. *For each $i \in \{0, \ldots, n-1\}$,*
     (a) *If $v^i = 0$,*
        i. *For each $j \in \{1, \ldots, m-1\}$, choose*

$$k_j^i \xleftarrow{\$} \mathbb{Z}_q$$
$$e_j^i \leftarrow \mathcal{H}(k_j^i + e_{j-1}^i m^i j H)$$

        taking $e_0^i = e_0$.

   ii. *Set* $C^i \leftarrow R^i / e^i_{m-1} = \frac{k^i_0}{e^i_{m-1}} G$.
   iii. *For each* $j \in \{1, \ldots, m-1\}$, *set* $s^i_j \leftarrow k^i_j + \frac{k^i_0 e^i_{j-1}}{e^i_{m-1}}$.
(b) *Otherwise,*
   i. *For each* $j \in \{1, \ldots, v^i - 1\}$, *choose* $s^i_j \xleftarrow{\$} \mathbb{Z}_q$, *and compute* e^i_j *directly from equation* (1), *taking* $e^i_0 = e_0$. *(If* $v_i = 1$ *this is a no-op.)*
   ii. *Set* $s^i_{v^i} = k^i + e^i_{v^i - 1} r^i$.

5. *Set* $C \leftarrow \sum_{i=0}^{n-1} C^i$. *Output*

$$\pi = \left\{ e_0, \left(C^0, s^0_1, s^0_2, \ldots, s^0_{m-1} \right), \ldots \left(C^{n-1}, s^{n-1}_1, s^{n-1}_2, \ldots, s^{n-1}_{m-1} \right) \right\}.$$

We observe that this is nearly the same construction as Borromean Ring Signatures except for the following two differences:

- There are no s^i_0 values, which were used in the calculation of $\hat{e}_0$ in the Borromean Ring Signature construction, saving i scalars in the total proof.
- The commitments C^i are no longer included in any hashes (which is necessary when computing sub-commitments to the digit $(m-1)$, as seen in step 4(a)ii of the Prove algorithm).
  Unfortunately, the resulting construction is no longer a secure ring signature in general; the proof of security depends on all keys being binding commitments rather than arbitrary public keys.

It is immediate that the above construction is a correct rangeproof. We argue security in the next two theorems.

Theorem 1. *If the underlying commitment scheme is binding in the sense of Definition 3, then the above construction is a proving rangeproof in the sense of Definition 7.*

Proof. Let (C, π) be generated by some p.p.t. algorithm $\mathcal{A}$, such that $\text{Verify}(C, \pi)$ accepts. Write

$$\pi = \left\{ e_0, \left(C^0, s^0_1, s^0_2, \ldots, s^0_{m-1} \right), \ldots \left(C^{n-1}, s^{n-1}_1, s^{n-1}_2, \ldots, s^{n-1}_{m-1} \right) \right\}.$$

By Theorem 8 in Appendix A, with nonnegligible probability $\mathcal{A}$ can be used to obtain openings (v^i, r^i) of each C^i with $v^i \in \{0, m^i, 2m^i, \ldots, (m-1)m^i\}$. By summing these we obtain an opening (v, r) of C with $v \in [0, m^n]$.

Theorem 2. *The above construction is zero-knowledge in the sense of Definition 8.*

Proof. This is nearly immediate. Observe that all values output by the Prove algorithm are selected independently uniformly at random, except where they are forced by the verification equations (which are independent of the committed values).

3.2 Confidential transactions

We now modify the definition of Bitcoin transaction (Definition 1).

Definition 10. *We define a* confidential transaction *as the following data:*

- *A list of* outputs*, containing a verification key, Pedersen commitment to an amount, and Back-Maxwell rangeproof that it lies in a range* $[0, 2^n - 1]$ *with* n *significantly smaller than the bit-length of size of the committed-value group.*
- *A list of* inputs*, which are unambiguous references to the outputs of other transactions, with signatures using the verification keys of those outputs.*
- *A* fee f*, which is listed explicitly.*

Our validity condition is changed as follows: the fee must be non-negative (except for coinbase transactions), the sum of all input commitments minus all output conditions must equal fH, and there must be valid signatures with all inputs' verification keys. This equation is important enough to give it a name.

Definition 11. *The* verification equation *is: input amounts minus output amounts equals fee (times* H*, when these amounts are considered as commitments).*

To summarize, the differences between confidential transactions and Bitcoin transactions are:

- Explicit amounts are replaced by homomorphically committed ones.
- Rather than computing the fee, it is given explicitly and checked that the inputs minus outputs commit to it.

Payment authorization is achieved by means of the input signatures, which are unchanged from Bitcoin and not discussed in this paper. However, we need to argue that this change does not allow coins to be created invalidly.

Theorem 3. *Consider a valid confidential transaction with fee* f*, inputs committing to amounts* $\{I_i\}_{i=0}^{k}$*, and outputs committing to amounts* $\{O_i\}_{i=0}^{\ell}$*. Suppose also that* $k + \ell + 1 < |\mathcal{C}|/R$*, where the output rangeproofs are to the range* $[0, R-1]$ *and* $f \in [0, R-1]$[2]*. If the rangeproofs are proving and the commitments are binding, then no subset of* $\{O_i\}$ *commits to more than* $\sum_{i=0}^{k} I_i - f$*.*

We observe that simply arguing that $\sum_{i=0}^{k} I_i - f = \sum_{i=0}^{\ell} O_i$ is insufficient: for example, with zero inputs and fee, an attacker could commit to two output amounts $\{1, -1\}$, and have created a coin from nowhere even though the total equation balances.

Proof. Since all rangeproofs are valid and commitments are binding, for each i we have $0 \leq O_i \leq R$. Similarly for the inputs, which are outputs of previous (valid) transactions.

[2] Typically the group order $\mathcal{C} \approx 2^{256}$ and $R \approx 2^{64}$ so this requirement is physically impossible to violate in practice.

Next, since the input commitments minus output commitments equal fH, we have $\sum_{i=0}^{k} I_i - f - \sum_{i=0}^{\ell} O_i \equiv 0 \pmod{|\mathcal{C}|}$, or

$$\sum_{i=0}^{k} I_i - f - \sum_{i=0}^{\ell} O_i = m|\mathcal{C}|$$

for some integer m. Now, since $k + \ell + 1 < |\mathcal{C}|/R$, and by our bounds on the individual terms, we can bound the left side of this equation as

$$-|\mathcal{C}| < \sum_{i=0}^{k} I_i - f - \sum_{i=0}^{\ell} O_i < |\mathcal{C}|.$$

But this implies that $m = 0$, *i.e.* that the input amounts add to the output amounts (plus fee).

Finally, since all output amounts are positive, every subset of outputs must sum to less than or equal to $\sum_{i=0}^{k} I_i - f$, as desired.

3.3 Performance

Consider a group $\mathcal{G}$ where both scalars and group elements are encoded in 1 unit of space (in practice, 32 bytes or 256 bits). We contrast three schemes: a naive folklore rangeproof using separate AOS ring signatures for each digit; one using Borromean Ring Signatures [15] as implemented in Elements Alpha[2]; and our scheme described above. We compare asymptotic and also look at the specific case $2^{38} \approx 3^{24}$. While the naive and Alpha schemes are space-optimal in base 4, our scheme is space-optimal in base 3. [3]

Scheme	Base	Digits	Range	Total Size
Naive	m	n	m^n	$(m+2)n$
Alpha	m	n	m^n	$1+(m+1)n$
Ours	m	n	m^n	$1+mn$
Naive	4	19	2^{38}	114
Alpha	4	19	2^{38}	96
Ours	3	24	3^{24}	73

For this range we observe a 24% reduction from the Alpha rangeproof and 36% reduction from the naive rangeproof at a slightly larger range.

[3] The optimality of one base over another comes from the fact that numbers in higher bases have fewer digits, reducing the size of each OR proof, while increasing the size of the individual OR proofs. Since in base b, the Alpha rangeproof requires b scalars and a commitment, while our optimization requires only $b-1$ scalars and a commitment, the optimum has shifted.

4 Confidential assets

4.1 Asset Commitments and Surjection Proofs

Before moving on, we need a few more primitives.

Definition 12. *Given some asset description A (whose precise form is given in Section 4.4), the associated* asset tag *is an element $H_A \in \mathcal{G}$ obtained by execution of the Pedersen commitment* Setup *using A as auxiliary input.*

When using multiple Pedersen commitment schemes, we distinguish them by adding their second generator as a subscript to their algorithms, like Open_{H_A} *or* Commit_{H_A}.

In particular, in the random oracle model an asset tag is a uniformly random curve point whose discrete logarithm is not known with respect to G or any other asset tag.

Definition 13. *Given an asset tag H_A, an* (ephemeral) asset commitment *is a point of the form $H = H_A + rG$, for uniformly random r. We sometimes abuse terminology to say that H is a* commitment to the asset *H_A.*

In the next section, we are going to use these asset commitments in place of the generator H in our Pedersen commitments. The following theorems justify this.

Theorem 4. *Let H be an asset commitment to asset tag H_A, and C a Pedersen commitment such that $\text{Open}_H(v, r, C)$ accepts. Then if $H_A = H + sG$, $\text{Open}_{H_A}(v, r - sv, C)$ accepts.*

This theorem is immediate, and implies that Pedersen commitments to an amount with some asset commitment as generator are also Pedersen commitments to the same amount with the underlying asset tag as generator. Further, anyone who knows the blinding factor s and the opening information with respect to one generator can determine the opening information with respect to the other generator.

Such Pedersen commitments commit not only to the committed amount, but also to the underlying asset tag, in the following sense.

Theorem 5. *If a p.p.t. algorithm $\mathcal{A}$ exists which can win with nonnegligible probability in the following game, then a simulator $\mathcal{B}$ exists which can solve the discrete logarithm problem for $\mathcal{G}$ with nonnegligible probability.*

1. *$\mathcal{A}$ calls* Setup_i *to produce asset tags H_i for $i = 0, 1, \ldots, n$.*
2. *$\mathcal{A}$ produces commitments C_i and openings (v_i, r_i) such that* $\text{Open}_i(v_i, r_i, C_i)$ *accepts for $i = 1, \ldots, n$.*
3. *$\mathcal{A}$ produces an opening (v, r) such that $v \neq 0$ and* $\text{Open}_0\left(v, r, \sum_{i=1}^{n} C_i\right)$ *accepts.*

The proof of this theorem is given in the Appendix.

By the comment following Definition 7, the same theorem holds if the adversary is required to produce rangeproofs rather than opening information.

We will attach fresh random asset commitments to all transaction outputs, and we need a way to link inputs and outputs without revealing the mapping. The following tool will be essential.

Definition 14. *An* asset surjection proof *(ASP) scheme consists of the following algorithms.*

- Prove *takes a collection* $\{H_i\}_{i=1}^n$ *of "input" asset commitments, an "output" commitment* $H = H_{i^*} + rG$ *for some* $1 \leq i^* \leq n$*, and* r*. It outputs a proof* π*.*
- Verify *takes a collection* $\{H_i\}_{i=1}^n$*,* H*, and a proof* π *and either accepts or rejects.*

We often say that an ASP is *from* the set $\{H_i\}$ of input commitments *to* the output commitment H.

Definition 15. *An ASP is* secure *if a proof* π *produced by the* Prove *algorithm is a zero-knowledge proof of knowledge (zkPoK) of the blinding factor* r*.*

This is easy to construct from a ring signature which is a zkPoK of one of its secret keys, for example the AOS ring signatures described in [1].

Definition 16. *The* AOS ASP *is the following:*

- Prove *computes the* n *differences* $H - H_i$ *for* $i = 1, \ldots, n$ *(one of which will be* r*) and computes a ring signature of an empty message with these differences. The proof* π *is the signature.*
- Verify *computes the same differences and verifies the ring signature.*

It is immediate that the AOS ASP is secure if the underlying AOS ring signature scheme is a zkPoK.

4.2 Confidential assets

Up to now, we have considered a single asset (for example Bitcoin) and transactions which move this asset from one holder to another. Consider an extension of this scheme which supports multiple non-interchangeable *asset types* (for example, BTC and a USD proxy) within single transactions. This increases the value of the chain by allowing it to serve more users, and also enables new functionality, such as atomic trades of different assets.

We could accomplish this by attaching to each output an *asset tag* identifying the type of that asset, and having verifiers check that the verification equation holds for subsets of the transaction which have only a single asset type. (Basically, treating the transaction as multiple single-asset transactions, except that each input signs the entire aggregate transaction.)

This requires verification of multiple equations, increases complexity, and more importantly, gives chain analysts an additional data point to consider, reducing the privacy of the users of the chain. This also could lead to censorship of transactions involving specific asset types, since all asset types are visible.

We instead propose a scheme for which all asset tags are blinded, so that no relationship between output asset types can be inferred. This avoids the privacy loss and greatly *improves* privacy by hiding the specific assets used by individual transactions. This is especially important for assets with low transaction volume where use of the asset alone is sufficient to identify users.

Definition 17. *A* confidential asset transaction *is the following data:*

- *A list of* inputs*, which are one of two forms:*
  - *an unambiguous reference to an output of another transaction, with a signature using that output's verification key*
  - *an* asset issuance input*, which has an explicit amount and asset tag; the precise validity rules for these are defined outside of this paper, but they are discussed further in Section 4.4.*
- *A list of* outputs*, containing*
  - *a verification key,*
  - *an asset commitment H_o with a ASP from all input asset commitments to H_o;*
  - *Pedersen commitment to an amount using generator H_o in place of H, with Back-Maxwell rangeproof (also using H_o in place of H) that it lies in a range $[0, 2^n - 1]$ with n significantly smaller than the bit-length of size of the committed-value group.*
- *A* fee *$\{(f_i, H_i)\}_{i=1}^n$, which is listed explicitly. Here the f_i's are scalar amounts denominated in the assets whose tags are the respective H_i. We require all H_i's to be distinct for simplicity. (Note that the asset types used to pay fees must be revealed. In practice we expect a working system to use fees denominated in only one asset, say, Bitcoin, so privacy is not lost.)*
  Each f_i must always be nonnegative; assets originate in asset-issuance inputs, which take the place of coinbase transactions in confidential transactions.

The validity equation is identical to that for confidential transactions, except that the fee commitment is calculated as $\sum_{i=1}^n f_i H_i$ instead of simply fH.

Again, payment authorization is achieved by means of the input signatures, so we do not argue this, only that no assets are created. We first prove a theorem to argue that the construction is sensible.

Theorem 6. *Consider a valid confidential asset transaction and let H be any fixed asset tag. Then the transaction is* valid for *H, in the following sense. Restrict the transaction to those inputs and outputs whose asset commitments are to H, and take $f = f_i$ if $H_i = H$ for any i and zero otherwise.*

Then if the discrete logarithm problem is hard in the underlying group, the sum of input commitments minus the sum of output commitments of this restricted transaction cannot be opened to any amount except f.

Proof. Consider the algorithm $\mathcal{A}$ which produced the transaction, and the transactions whose outputs are used as inputs, and so on. (In practice $\mathcal{A}$ will be the conjunction of many different transacting parties, but this does not affect our argument.)

Since every output has a rangeproof and ASP associated to it, which are proofs of knowledge of the opening information and asset commitment blinding factor, respectively, of every output, there exists a simulator $\mathcal{B}$ which extracts this information from $\mathcal{A}$. Using the blinding factors and Theorem 4, we can consider every rangeproof as being with respect to the underlying asset tag, rather than the asset commitment.

Now, consider the sum of the outputs minus inputs minus fH of the restricted transaction is some commitment C. This commits to some amount of H. But since the non-restricted transaction is valid, we have that the remaining outputs minus inputs, minus remaining fees, equals $-C$. Since the remaining inputs, outputs, and fees are commitments to non-H asset tags, by Theorem 5, C must commit to 0, completing the proof.

Theorem 7. *Consider a valid confidential asset transaction and let H be any fixed asset tag. Suppose the transaction has fee f $\{(f_i, H_i)\}_{i=1}^{n}$, inputs committing to amounts $\{I_i\}_{i=0}^{k}$, and outputs committing to amounts $\{O_i\}_{i=0}^{\ell}$. Suppose also that $k + \ell < |\mathcal{C}|/R$, where the rangeproofs prove to the range $[0, R-1]$. If the rangeproofs are proving and the commitments are binding, then no subset of $\{O_i\}$ commits to more than $\sum_{i=0}^{k} I_i - f$.*

Proof. By the above theorem, the transaction restricted to only inputs and outputs with asset tag H is a valid confidential transaction, except that the output commitments minus input commitments minus fee sum to a commitment to 0, rather than the 0 point itself. The proof of Theorem 3, which does not make use of this distinction, therefore goes through without change on the restricted transaction.

4.3 Performance

In Section 3.3 we described the size of our rangeproofs, which are attached to every transaction output. This is unchanged for confidential assets, but we also require two additional pieces of data: an asset commitment and an ASP showing that this commitment is legitimate.

In the units of Section 3.3, the asset commitment has size 1 and the ASP has size $n + 1$, where n is the number of inputs that a given output may have come from.

For any entire transaction with m outputs and n inputs, the additional data therefore has size $m(n + 2)$. We can improve this at the cost of privacy by using a weaker form of an ASP which proves an asset commitment is the same as one of 3 inputs, rather than being the same as any of them. The additional data would then have cost only $5m$, which is asymptotically better.

4.4 Issuance

As discussed in Section 4.1, the *asset tag* is an element $H_A \in \mathcal{G}$ obtained by execution of the Pedersen commitment Setup using an auxiliary input A. In the context of a blockchain, we want to ensure that any input A is used only once to ensure assets cannot be inflated by means of multiple independent issuances. Associating an issuance with the spend of a UTXO, and a maximum of one issuance per specific UTXO achieves this uniqueness property. The unambiguous reference to the UTXO being spent is hashed together with a issuer-specified value, the *Ricardian contract hash*[9], to generate the auxiliary input A to the Pedersen commitment.

Definition 18. *Given an input being spent I, itself an unambiguous reference to an output of another transaction, and the issuer-specified* Ricardian contract *C, the* asset entropy *E is defined as $Hash(Hash(I)||Hash(C))$.*

The *Ricardian contract* is a machine parseable legal document specifying the conditions for use, and especially redemption of the asset being issued [9]. The details of how such a contract might be designed or enforced is outside the scope of this paper. All that matters for the purposes here is that such a document exists and that its hash is irrevocably committed to in the issuance of the asset.

Definition 19. *Given an* asset entropy *E, the* asset tag *is the element $H_A \in \mathcal{G}$ obtained by execution of the Pedersen commitment* Setup *using $Hash(E||0)$ as the auxiliary input.*

Every non-coinbase transaction input can have associated with it up to one new asset issuance:

Definition 20. *An* asset issuance input *consists of an UTXO spend I (interpreted as a non-issuance input of the same transaction); a* Ricardian contract *C; an initial issuance explicit value v_0, or Pedersen commitment H and Back-Maxwell rangeproof P_0; and a Boolean field indicating whether reissuance is allowed.*

Reissuance will be explained in Section 4.5.

4.5 Reissuance and capability tokens

Assets may be either of fixed issuance or, optionally, enable later reissuance using a *asset reissuance capability*. This capability is a token providing its owner with the ability to change the amount of asset in circulation at any point after the initial issuance. When a reissuable asset is created, both the initial asset issuance and the reissuance capability token are generated at the same time.

Definition 21. *Given an* asset entropy *E, the* asset reissuance capability *is the element $H_A \in \mathcal{G}$ obtained by execution of the Pedersen commitment* Setup *using $Hash(E||1)$ as the auxiliary input.*

An asset which supports reissuance indicates this in its asset issuance input, and the transaction contains an additional output of amount 1 which commits to asset tag H_A.

Note the parallel to the definition of the *asset tag* given in Section 4.4, but with the concatenation of a different constant before hashing. In this way an asset tag is linked to its corresponding reissuance capability, and the holder of such a capability is able to assert their reissuance right simply by revealing the blinding factor for the capability along with the original asset entropy.

Definition 22. *An* asset reissuance input *consists of a spend of a UTXO containing an asset reissuance capability; the original asset entropy E; the blinding factor for the asset commitment of the UTXO being spent; and either an explicit reissuance amount v_i, or Pedersen commitment H and Back-Maxwell rangeproof P_i.*

We call attention to the fact that this reissuance mechanism is a specific instance of a general capability-based authentication scheme. It is possible to use the same scheme to define capabilities that gate access to other restricted operations. In the authors' implementation there exists separate capabilities for increasing and decreasing issuance, and explicit vs committed reissuance amounts. In general the right being protected could even be made extensible by making the commitment generator the hash of a script that validates the spending transaction.

4.6 Performance

In contrast to Confidential Transactions, in which every output has an attached rangeproof, each Confidential Assets output must also have an asset tag and asset surjection proof. As in Section 3.3, we consider curvepoints and scalars to have the same size,

For an output whose amount is in the range $[0, m^n)$ and whose asset references A assets, the total size of of the rangeproof and ASP is therefore $(1+mn)+(2+A)$ where the first term is the contribution of the rangeproof and the second the contribution of the asset tag and ASP. For a prototypical example of a $[0, 3^{24})$ rangeproof and three inputs, the total is 78 scalars, or 19968 bits.

4.7 "Small Assets" and "Big Assets"

To prove that the asset commitments associated to outputs commit to legitimately issued asset tags, we have used asset surjection proofs which show that they commit to the same asset tag as some input (if those inputs are outputs of previous transactions, they have ASP's showing the same thing, and so on until the process terminates at an asset issuance input which has an explicit asset tag).

This allows confidential assets to work on a blockchain which supports indefinitely many asset types, which may be added after the chain has been defined.

An alternate scheme, which works for a small fixed set of asset tags, is to define the asset tags at the start of the chain, and to have each output include an ASP to the global list of asset tags. We refer to this scheme as "small assets" and the more general scheme as "big assets".

It is also possible to do an intermediate scheme, by having a global dynamic list of assets with each transaction selecting a subset of asset tags which its outputs have an ASP to. In general, there is room to adapt this scheme for optimal tradeoff between ASP size and privacy for specific use cases.

We observe that small assets is compatible with Mimblewimble [11], a new extension to confidential transactions which improves privacy and scaling by removing information from the transaction graph, while big assets is not.

5 Future Research

The authors describe some research directions they would like to see.

Rangeproof Efficiency. While this paper describes the most efficient rangeproof construction without trusted setup that the authors are aware of, in practice for a blockchain-based currency, rangeproofs are still the bulk of the transaction data. Further improvements, especially asymptotic ones, would help.

ASP Efficiency. Similarly, the ASP construction scales with both the number of inputs and the number of outputs; by restricting the set of inputs it uses we improve this at cost of user privacy, but it is desirable to avoid this tradeoff.

Aggregate Rangeproofs. If it were possible to aggregate rangeproofs (*e.g.* to combine proofs that C_1 and C_2 commit to values in $[0, 2^n - 1]$ into a single proof that $C_1 + C_2$ commits to a value in $[0, 2^{n+1} - 1]$, this would also improve the efficiency of a blockchain-based system, since proofs could be placed in a Merkle-sum tree whose nodes contained an aggregate rangeproof of the rangeproofs of their children. Then validators could check only the root to ensure an entire tree did not cause any inflation, delaying checking the proofs on individual outputs until those outputs are spent.

Quantum Resistance. The primitives described in this paper all depend on the elliptic-curve discrete logarithm assumption, which is known to be insecure against a quantum adversary. A quantum-hard analogue would require a replacement for Pedersen commitments (perhaps [4]), for the ring signatures used by ASP's, and for rangeproofs.

A Appendix: Proofs

Theorem 8. *Fix integers $i \geq 0$, $m > 0$. Consider an algorithm $\mathcal{A}$ which can produce the tuple*

$$\pi = (\alpha, e_0, C, s_1, \ldots, s_{m-1})$$

such that one can define, for $j \in \{1, \ldots, m-1\}$,

$$e_j \leftarrow \mathcal{H}\left(s_j G - e_{j-1}\left[C - jm^i H\right]\right),$$

$$R \leftarrow e_{m-1} C,$$

and it holds that $e_0 = \mathcal{H}(R \| \alpha)$. (Observe that the formula for e_j is the same as (1) from Definition 9; this represents the verification equation of a single ring. Here α is auxiliary data that $\mathcal{A}$ chooses, but in the full algorithm it consists of the R values from the other rings.)

Then a simulator $\mathcal{B}$ exists, which given oracle access to $\mathcal{A}$, can extract an opening (v, r) such that $\mathrm{Open}(v, r, C)$ *accepts and $v \in \{0, m^i, \ldots, (m-1)m^i\}$.*

Proof. Suppose that $\mathcal{A}$ makes at most q random oracle queries. $\mathcal{B}$ acts as follows. For each random oracle query it chooses a uniformly random scalar and responds with this.

It chooses $i^* \in \{1, \ldots, q\}$ uniformly at random, and on the i^*th query, $\mathcal{B}$ forks $\mathcal{A}$ into $\mathcal{A}$ and $\mathcal{A}'$. It gives e_{i^*} to $\mathcal{A}$, e'_{i^*} to $\mathcal{A}'$, and answers further queries from other algorithms with uniformly random values.

Let the final output of the two algorithms be

$$\pi = (\alpha, e_0, C, s_1, \ldots, s_{m-1})$$

$$\pi' = (\alpha', e'_0, C', s'_1, \ldots, s'_{m-1})$$

and similarly e_j and e'_j are defined as in the hypothesis.

With probability $1/q - negl$, we have $e_j = e'_j$ for all j except one, j^*. (This is the probability that the i^*th query was the last e_j that $\mathcal{A}$ needed, and that it obtained every e_j by querying the random oracle rather than guessing.) Abort otherwise.

We consider four cases.

1. If $j^* = m - 1$, then

   $$e_0 = \mathcal{H}(e_{m-1} C \| \alpha) = \mathcal{H}(e'_{m-1} C' \| \alpha') = e'_0$$

   so that except with negligible probability, $\alpha = \alpha'$ and $C = \frac{e'_{m-1}}{e_{m-1}} C'$. Now,

   $$e_{m-1} = \mathcal{H}\left(s_{m-1} G - e_{m-2}\left[C - (m-1)m^i H\right]\right)$$

   $$e'_{m-1} = \mathcal{H}'\left(s'_{m-1} G - e_{m-2}\left[C' - (m-1)m^i H\right]\right)$$

where $\mathcal{H}$, $\mathcal{H}$' are used to emphasize which side of the fork received these random oracle responses. But by hypothesis, the input to these queries is the same, that is,

$$s_{m-1}G - e_{m-2}\left[C - (m-1)m^iH\right] = s'_{m-1}G - e_{m-2}\left[C' - (m-1)m^iH\right]$$

which is sufficient to solve for the discrete logarithms r, r' of $C-(m-1)m^iH$ and $C' - (m-1)m^iH$, giving us openings $(m-1, r)$ and $(m-1, r')$ for the commitments of the two forks.

2. If $j^* \neq m-1$ and $C = C'$,then

$$\begin{aligned} e_{j^*+1} &= \mathcal{H}\left(s_{j^*+1}G - e_{j^*}\left[C - j^*m^iH\right]\right) \\ &= \mathcal{H}\left(s'_{j^*+1}G - e'_{j^*}\left[C - j^*m^iH\right]\right) \\ &= e'_{j^*+1} \end{aligned}$$

and we can solve for the discrete logarithm r of $C - j^*m^iH$, and our desired opening for C (the output of both forks) is (j^*m^i, r).

3. If $j^* = 0$ and $C \neq C'$, we have that the inputs to

$$e_0 = \mathcal{H}(e_{m-1}C \| \alpha)$$

$$e'_0 = \mathcal{H}(e'_{m-1}C' \| \alpha')$$

are the same, and $e_{m-1} = e'_{m-1}$ by hypothesis. This implies $C = C'$, a contradiction.

4. If $0 < j^* < m-1$ and $C \neq C'$, observe that

$$e_{j^*} = \mathcal{H}\left(s_{j^*}G - e_{j^*}\left[C - j^*m^iH\right]\right)$$

$$e'_{j^*} = \mathcal{H}'\left(s'_{j^*}G - e'_{j^*}\left[C' - j^*m^iH\right]\right)$$

and as in case 1, by hypothesis

$$s_{j^*}G - e_{j^*}\left[C - j^*m^iH\right] = s'_{j^*}G - e'_{j^*}\left[C' - j^*m^iH\right] \tag{2}$$

Similarly,

$$\begin{aligned} e_{m-1} &= \mathcal{H}\left(s_{m-1}G - e_{m-2}\left[C - (m-1)m^iH\right]\right) \\ &= \mathcal{H}\left(s'_{m-1}G - e'_{m-2}\left[C' - (m-1)m^iH\right]\right) \\ &= e'_{m-1} \end{aligned}$$

so

$$s_{m-1}G - e_{m-1}\left[C - (m-1)m^iH\right] = s'_{m-1}G - e'_{m-1}\left[C' - (m-1)m^iH\right] \tag{3}$$

Now, after rearranging, (2) is

$$\frac{1}{j^*m^i(e'_{j^*} - e_{j^*})}\left[(s_{j^*} - s'_{j^*})G + e'_{j^*}C' - e_{j^*}C\right] = H$$

and (3) is

$$\frac{1}{(m-1)m^i(e'_{m-1}-e_{m-1})}\left[(s_{m-1}-s'_{m-1})G+e'_{m-1}C'-e_{m-1}C\right]=H$$

which combine to determine the discrete logarithms r, r' of C and C', so that $(0,r)$ and $(0,r')$ are the desired openings.

A.1 Proof of Theorem 3

Proof. Recall that G is a fixed random generator of $\mathcal{G}$. Let (G,X) be $\mathcal{B}$'s discrete logarithm challenge, *i.e.* $\mathcal{B}$ succeeds if it outputs x such that $X=xG$. We consider two types of adversary: a type I adversary's output satisfies $\sum_{i=1}^n r_i \neq r$, while a type II has equality. We assume that $\mathcal{A}$ makes at most q random oracle queries.

For a Type I adversary, $\mathcal{B}$ acts as follows.

First, $\mathcal{B}$ responds to random oracle queries by choosing random scalars r and replying with rX. Then from $\mathcal{A}$'s perspective, Setup_i outputs uniformly a random generators H_i; however $\mathcal{B}$ knows scalars s_i such that $H_i=s_iX$.

Now, let (C_i,v_i,r_i,v,r) for $i=1,\ldots,n$ be the output of $\mathcal{A}$. Write $C=\sum_{i=1}^n C_i$. We have

$$\begin{aligned}
0 &= C-\sum_{i=1}^n C_i \\
&= vH_0+rG-\sum_{i=1}^n[v_iH_i+r_iG] \\
&= vs_0X+rG-\sum_{i=1}^n[v_is_iX+r_iG] \\
&= \left[vs_0-\sum_{i=1}^n v_is_i\right]X+\left[r-\sum_{i=1}^n r_i\right]G
\end{aligned}$$

Since the sum in the right term is nonzero for a type I adversary, so must be the sum in the left term, so we have

$$x=\frac{r-\sum_{i=1}^n r_i}{vs_0-\sum_{i=1}^n v_is_i}$$

which satisfies $X=xG$.

For a Type II adversary, $\mathcal{B}$ acts as follows. It responds for the Type I simulator, except for one random oracle queries it replies with sG rather than sX. Then with probability $1/q$ we have $H_0=s_0G$, and if not we abort. We also abort if $s_0=0$, which occurs with negligible probability.

The above equation then becomes

$$0=\left[\sum_{i=1}^n v_is_i\right]X+\left[vs_0+r-\sum_{i=1}^n r_i\right]G$$

where the right term is equal to $vs_0 \neq 0$, so the left term must also be nonzero, and

$$x = \frac{vs_0}{\sum_{i=1}^{n} v_i s_i}$$

satisfies $X = xG$.

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: In Advances in Cryptology - ASIACRYPT 2002, LNCS. pp. 415–432. Springer-Verlag (2002)
2. Back, A.: Announcing sidechain elements: Open source code and developer sidechains for advancing bitcoin (2015), Blockstream blog post, `https://blockstream.com/2015/06/08/714/`
3. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. Cryptology ePrint Archive, Report 2013/507 (2013), `http://eprint.iacr.org/2013/507`
4. Cabarcas, D., Demirel, D., Göpfert, F., Lancrenon, J., Wunderer, T.: An unconditionally hiding and long-term binding post-quantum commitment scheme. Cryptology ePrint Archive, Report 2015/628 (2015), `http://eprint.iacr.org/2015/628`
5. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: ASIACRYPT 2008. pp. 234–252 (2008)
6. Chaabouni, R., Lipmaa, H., Zhang, B.: A non-interactive range proof with constant communication. In: Financial Cryptography 2012. pp. 179–199 (2012)
7. Fouque, P.A., Tibouchi, M.: Indifferentiable Hashing to Barreto–Naehrig Curves, pp. 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
8. Friedenbach, M., Timón, J.: Freimarkets: extending bitcoin protocol with user-specified bearer instruments, peer-to-peer exchange, off-chain accounting, auctions, derivatives and transitive transactions (2013), `http://freico.in/docs/freimarkets-v0.0.1.pdf`
9. Grigg, I.: The ricardian contract. In: First IEEE International Workshop on Electronic Contracting. IEEE (2004)
10. Hearn, M.: Merge avoidance: Privacy enhancing techniques in the bitcoin protocol (2013), `http://www.coindesk.com/merge-avoidance-privacy-bitcoin/`
11. Jedusor, T.: Mimblewimble (2016), defunct hidden service, `http://5pdcbgndmprm4wud.onion/mimblewimble.txt`. Reddit discussion at `https://www.reddit.com/r/Bitcoin/comments/4vub3y/mimblewimble_noninteractive_coinjoin_and_better/`
12. jl2012: OP_CHECKCOLORVERIFY: soft-fork for native color coin support (2013), BitcoinTalk post, `https://bitcointalk.org/index.php?topic=253385.0`
13. Maxwell, G.: CoinJoin: Bitcoin privacy for the real world (2013), BitcoinTalk post, `https://bitcointalk.org/index.php?topic=279249.0`
14. Maxwell, G.: Confidential transactions (2015), plain text, `https://people.xiph.org/~greg/confidential_values.txt`
15. Maxwell, G., Poelstra, A.: Borromean ring signatures (2015), `http://diyhpl.us/~bryan/papers2/bitcoin/Borromean%20ring%20signatures.pdf`
16. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009), `https://www.bitcoin.org/bitcoin.pdf`

17. Pedersen, T.: Non-interactive and information-theoretic secure verifiable secret sharing. Lecture Notes in Computer Science 576, 129–140 (2001)
18. Project, E.: Create your own crypto-currency with ethereum (2016), `https://www.ethereum.org/token`. Retrieved on 2016-10-31.
19. Schoenmakers, B.: Interval proofs revisited (2005), slides presented at *International Workshop on Frontiers in Eleectronic Elections*
20. Southurst, J.: Blockchain's sharedcoin users can be identified, says security expert (2014), `http://www.coindesk.com/blockchains-sharedcoin-users-can-identified-says-security-expert/`
21. Wilcox-O'Hearn, Z.: Zcash begins (2016), zCash Blog Post, `https://z.cash/blog/zcash-begins.html`. Retrieved 2016-10-31.
22. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (2014), `http://gavwood.com/paper.pdf`