Zano: Confidential Assets Scheme for RingCT and Zarcanum

sowle[1],

[1]Zano project, val@zano.org

February 2024*

Abstract

In this paper, we describe a practical way of implementing confidential assets (a.k.a. tokens or colored coins) in Zano with unlimited decoy mixing capability and hidden amounts as an extension to the Ring Confidential Transactions scheme. Our approach preserves public verifiability that no transaction creates or destroys coins. We further extend this approach to show how it can be combined with Zarcanum, a Proof-of-Stake scheme for transaction with hidden amounts.

1 Introduction

The Zano project is heading towards a major privacy update, planned on March 2024. With this update and subsequent blockchain hard-fork all newly created transaction outputs will have hidden amounts, while it still be possible to stake them in a fully anonymous manner. At the same time, it will be possible to transfer multiple asset types within single transactions. In this paper we describe our approach, which is based on the concept of confidential assets with blinded assets tags originally proposed in [10]. We show how we retain public verifiability that no assets are created or destroyed, while hiding both the output amounts and the output asset types.

2 Notation

Let $\mathbb{G}$ denote the main subgroup of the Ed25519 curve and let $\mathbb{Z}_p$ denote a ring of integers modulo p.
Let l be the order of $\mathbb{G}$: $l = \#\mathbb{G} = 2^{252} + 27742317777372353535851937790883648493$.
For any set X, $x \xleftarrow{\$} X$ means uniform sampling of x at random from X.
H_s is a scalar hash-function: $H_s : \{0,1\}^* \to \mathbb{Z}_l$
H_p is a deterministic hash function: $H_p : \{0,1\}^* \to \mathbb{G}$.
$G, H, X, U \in \mathbb{G}$ are fixed group generators with no efficiently-computable discrete logarithm relations selected uniformly at random (unless stated otherwise).

3 Confidential Assets construction

3.1 Asset descriptor

Our approach is based on the concept of confidential assets with blinded assets tags originally proposed in [10] and adapted in [3]. Here we briefly describe the concept.

In a normal confidential transaction each output's amount a is committed to in amount commitment A using additively homomorphic Pedersen commitment (see also [9]):

$$A = aH + fG$$

where $f \in \mathbb{Z}_l$ is a random blinding factor.

*Version 1.0. Last update: 2024-02-03.

The idea is to use a unique amount-bonded generator H_t[1] for each asset t instead of the generator H. So the amount a for the asset t is committed to in

$$A_t = aH_t + fG$$

To publicly and unambiguously link an asset with the corresponding generator H_t, the latter is calculated as

$$H_t = H_p(asset\_descriptor\_t)$$

where $asset\_descriptor\_t$ is data structure (Fig. 1), containing the information associated with the given asset, such as unique name, ticker, emission parameters, etc. *Asset descriptor* is explicitly put into the blockchain during the initial emission of an asset, so that blockchain observers can calculate corresponding asset tag H_t and keep track of all existed asset types for further use.

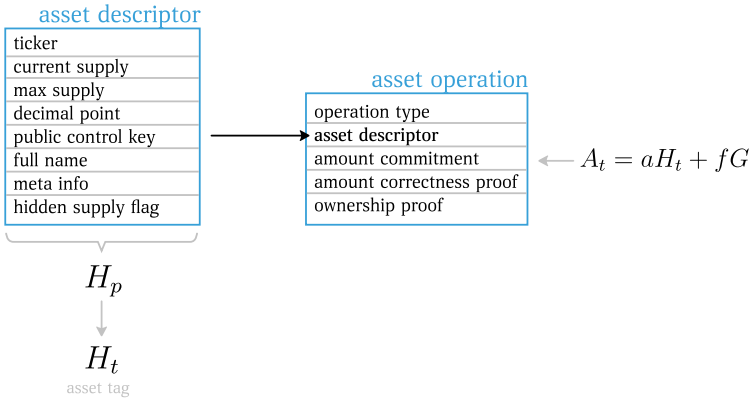

Fig. 1. Asset descriptor and asset operation data structures outline

Note, that $\{H_t\}$ should have no efficient-computable discrete logarithm relations with themselves and other generators.

For native Zano coins we use predefined asset tag $H_{native} = H$.

3.2 Hiding the real asset

Outputs' asset tags are necessary for verification. However, instead of disclosing asset tag H_t for each output, we disclose the blinded asset tag T, because otherwise it would be possible for observers to learn the type of a corresponding asset from it.

Blinded asset tag T is calculated as

$$T = H_t + sX = H_p(asset\_descriptor\_t) + sX$$

where $s = H_s(...)$ is a pseudo-random mask, known only to the output's owner and the sender.

3.3 Transaction structure

The structure of a typical transaction with confidential assets support is shown on Fig. 2. It has m inputs, each referring to a ring of size n, and k outputs. The index of a real output is denoted by $\pi \in [0, n)$ (assuming, it's distinct for each input).

Note, that each transaction's input can be either one of two possible types: 1) new *ZC*-input[2] with hidden amount and asset type, or 2) old so-called *bare input* with explicit amount and no asset support, as used in the original CryptoNote protocol [11]. ZC-inputs can only refer to ZC-outputs in their rings, and bare inputs can only refer to old CryptoNote-style outputs with explicit amounts. While each bare input requires an NLSAG signature, as in original CryptoNote, each ZC-input requires $^{d}/_{v}$-CLSAG[3] signature.

Public verifiability that no assets are created or destroyed, while hiding both the output amounts and the output asset types, is retained with the combination of balance proof, range proofs and asset tags surjection proof which will be discussed below.

[1] In [10] such generators are called *asset tags*. In this paper we use terms *asset tag* and *asset id* interchangeably.

[2] ZC signature stands for Zarcanum-aware CLSAG signature. The same acronim is used for corresponding inputs and outputs.

[3] $^{d}/_{v}$-CLSAG is the CLSAG[5] extension described in [12].

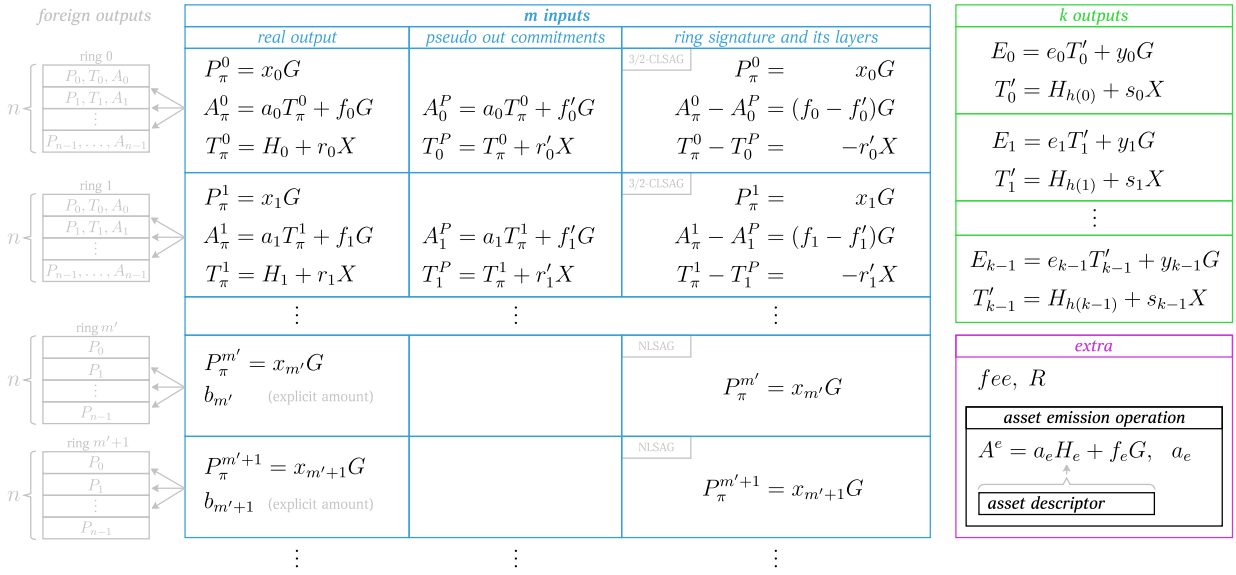

Fig. 2. Transaction with Confidential Assets, employing both ZC inputs and old bare inputs

3.4 Ring signature construction

For ZC-inputs we use RingCT approach ([7], [8]) of using pseudo output commitment to implement simple balance proof, while hiding the commitment of the real output. In addition to pseudo output amount commitment $A_i^P = a_i T_\pi^i + f_i' G$ we use pseudo output blinded asset tag $T_i^P = T_\pi^i + r_i' X$, where f_i', r_i' are pseudo-random blinding masks.

To prove that T^P corresponds to ring asset tag T_π with the same index π as stealth address P_π and amount commitment A_π we use additional layer of the ring signature:

$$T_\pi - T^P = -r' X$$

For the ring signature itself we use $^d/_v$-CLSAG ([12]) instead of standard d-CLSAG signature, because this additional layer employs different group generator (X). Namely, we use $^3/_2$-CLSAG: two generators (G, X) in a 3-layer arrangement (G, G, X).

3.5 Asset registration, emission and public burning

All asset operations are done by sending a transaction with *asset operation* (Fig. 1) data structure in its extra section. Registered asset can be controlled by *master asset secret control key*, and all operations, except the registration, require the asset operation structure to be signed by asset control key. Blockchain observers then can verify this.

To emit an asset one fills the mentioned data structures in transaction's extra to disclose emission amount a_e, secretly calculates blinding mask $f_e = H_s(domain\_sep, mck, R)$, where mck is the secret control key, and calculates corresponding amount commitment:

$$A^e = a_e H_e + f_e G$$

To complete asset emitting transaction, one or several outputs with blinded asset id corresponding to H_e and with the total amount of a_e should be added.

Similarly, to publicly burn a_b amount of an asset with tag H_b, one does the same calculations:

$$A^b = a_b H_b + f_b G$$

Unlike the emission of an asset, for public burn operation to be balanced, one or several inputs, corresponding to asset tag H_b and with the total amount of a_b, should be added.

Asset registration operation can at the same time fully or partially emit the supply of the asset.

3.6 Balance proof

For each distinct asset t corresponding amounts in inputs and outputs should be balanced:

$$\sum_{i:H_i=H_t} a_i = \sum_{i:H_{h(i)}=H_t} e_i \tag{1}$$

where mapping $h : [0, k) \to [0, m)$ maps indices of asset tags in outputs to indices of corresponding asset tags in inputs.

Similarly, amounts of native coins should be balanced, including the transaction fee[4] and bare inputs:

$$\sum_{i:H_i=H} a_i + \sum_j b_j = \sum_{i:H_{h(i)}=H} e_i + fee \tag{2}$$

Thanks to homomorphism we can combine balance equations (1) and (2) into one using corresponding commitments:

$$\Big(\sum_j b_j - fee\Big) \cdot H + \sum_{i=0}^{m-1} A_i^P - \sum_{j=0}^{k-1} E_j = sX \tag{3}$$

To bring the balance equation to the most general form we need to include amount commitments for asset emission (A^e) or for asset burning (A^b)[5]:

$$\Big(\sum_j b_j - fee\Big) \cdot H + \sum_{i=0}^{m-1} A_i^P + A^e - A^b - \sum_{j=0}^{k-1} E_j = sX \tag{4}$$

Observers make sure that the equation above holds for some secret s using a Schnorr proof. Blinding masks s_j and y_j in outputs are calculated using a shared secret[6] so output's receiver is able to reconstruct them.

To zero G-component in the left part of Eq. (5) we adjust one of blinding masks of pseudo output amount commitment, f'_j.

3.7 Asset tags surjection proof

Blinded asset tags T'_j in outputs have to be restricted by additional proof to prevent malicious use. For that purpose we use asset surjection proof scheme from [10], later improved in [3] with the help of logarithmic membership proof by Groth, Bootle at al. [6][2][1]. We also use the optimization, proposed in Section 1.3 in [4].

Given the set of pseudo output asset tags $\{T_i^p\},\ i = 0 \dots m-1$ we prove that each output's asset tag $\{T'_j\},\ j = 0 \dots k-1$ corresponds to one of them, i.e. we prove knowing a DL secret x with respect to X for one of a public key in the set $\{T_i^p - T'_j\}$:

$$T_i^p - T'_j = xX$$

(where m is the number of inputs and k is the number of outputs).

According to [1], communication costs can be estimated as $4.1\sqrt{k \log m}$ group elements and the same amount of field elements. Number of group exponentiations on verification is $O(\sqrt{k} \log km)$

In contrast, using the simplest case of non-aggregated ring signature would require $km+1$ field elements (k ring signatures, each having a ring of size m and a shared Fiat-Shamir challenge).

3.8 Range proof aggregation

Using different generators T'_j for each output commitment requires additional steps to aggregate range proofs (otherwise we would need to use single range proof for each output which is very consuming). For each output commitment $E_j = e_j T'_j + y_j G$ we provide additional commitment to the same amount $E'_j = e_j U + y'_j G$ (where $U \in \mathbb{G}$ is another fixed generator with no efficiently-computable discrete logarithm relation with others), and a vector Schnorr proof of knowing DL e_j, y'' in

$$E_j + E'_j = e_j(T'_j + U) + y''G$$

which in total would require 1 group element E'_j and 2 scalars per output, and one scalar for a shared Fiat-Shamir challenge and one aggregated range proof for all outputs.

[4] Here we assume that the transaction fee can only be paid in native coins, and its value is explicitly stated in transaction's extra.

[5] We assume that only one asset operation can be put into a transaction, hence either A^e or A^b may present in this equation, but not both.

[6] Namely, as $H_s(domain\_sep, H_s(8vR, i))$, where $domain\_sep$ is domain separation constant, v is recipient's secret view key, R is transaction public key, and i is output's index.

3.9 Transactions with no ZC inputs and implications

Let's consider an important[7] case when a transaction has only m bare inputs and zero ZC inputs.

In such a case there's no reason to hide real asset id in outputs using a non-zero blinding mask s_j, because it's obvious that all of them are $H_{native} = H$. Thus, $s_j = 0$ and the balance equation has zero X component.

Also, in the absence of ZC inputs we cannot zero G component by adjusting pseudo output amount blinding mask f'_j. This also means, that we cannot do any asset operation within such a transaction.

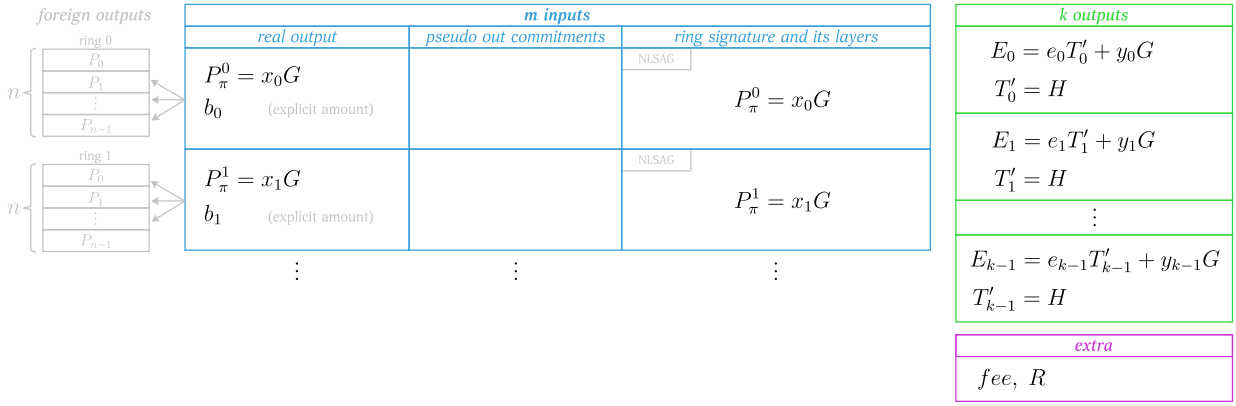

Fig. 3. Transaction with Confidential Assets, employing only CryptoNote-style bare inputs with explicit amounts

Considering all of mentioned above, we finally got simplified balance proof equation:

$$\left(\sum_j b_j - fee\right) \cdot H - \sum_{j=0}^{k-1} E_j = x_g G \tag{5}$$

As in the previous case, observers make sure that the equation above holds for some secret x_g using a Schnorr proof. Blinding masks y_j in outputs are calculated using a shared secret[8] so output's receiver is able to reconstruct them.

Note, that such a transaction generates ZC outputs with explicit asset id. If another transaction will spend only such outputs with explicit native coins asset id (with or without any bare inputs) it will mean that all outputs of such a transaction have native coin asset id as well. We detect such scenarios and enforce using explicit asset id for such obvious outputs with consensus-level rules, because we believe that eventually it will help improving the privacy.

3.10 Zarcanum Proof-of-Stake mining transaction

In this section we discuss how Zarcanum Proof-of-Stake scheme for confidential transactions, suggested in [13], can be implemented for transactions with confidential assets. Fig. 4 shows the simplest Zarcanum transaction, supporting Confidential Assets. We assume that only native coins can be staked to mine a PoS block.

Let us recall Zarcanum's main concept. A staker periodically checks all his unspent outputs in the wallet against PoS win condition. Once it satisfied, staker has an opportunity to create a PoS block, sign its miner transaction with Zarcanum signature, broadcast it and receive the block reward.

Zarcanum signature requires adding two additional layers to the ring signature:

1. a proof of knowing the DL of $C - A_\pi - Q_\pi$ with respect to X;
2. a proof of knowing the DL of Q_π with respect to G;

where C is specially constructed extended amount commitment (please, refer to Zarcanum whitepaper [13] for more details). Here we effectively have 5-layer (G, G, X, X, G) arrangement for the ring signature,

[7] In Zano ZC outputs are allowed only after Zarcanum hardfork, hence at the very moment of the hardfork there won't be a single ZC output in the blockchain.

[8] Namely, as $H_s(domain\_sep, H_s(8vR, i))$, where $domain\_sep$ is domain separation constant, v is recipient's secret view key, R is transaction public key, and i is output's index.

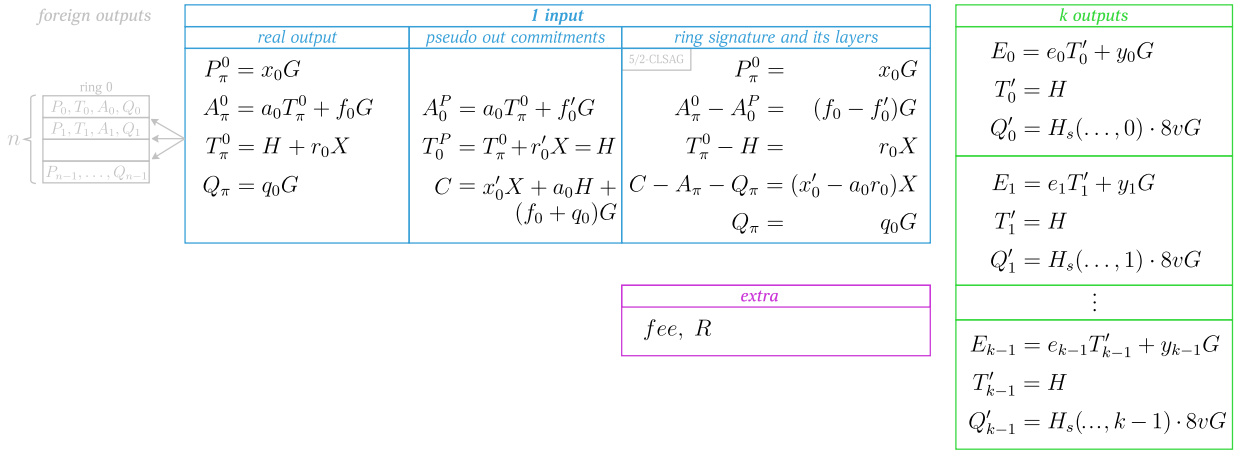

Fig. 4. The simplest Zarcanum PoS mining transaction with Confidential Assets support

while only two distinct generators are used. Thus, it's reasonable to employ $^d/_v$-CLSAG signature, namely, $^5/_2$-CLSAG ([12]).

Additional group element Q, which is calculated for each ZC output, is necessary to preserve sender-receiver anonymity in Zarcanum.

Note, that all outputs of the transaction on Fig. 4 have explicit asset id, because there's no other inputs and staking is only allowed for native coins. Using such PoS miner transactions may negatively impact blockchain anonymity, because it spends a stake output with a possibly non-explicit asset id ($r_0 \neq 0$), but instead creates one or several outputs with explicit asset id. This problem can be solved by adding an arbitrary ZC-input with non-explicit asset id to PoS mining transaction, as shown on Fig. 5.

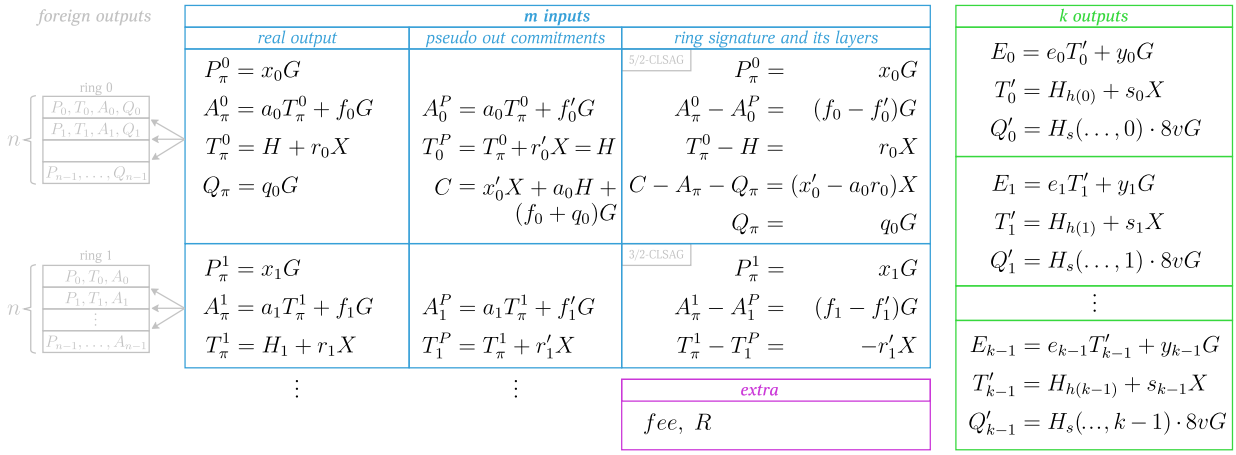

Fig. 5. Zarcanum PoS mining transaction with Confidential Assets support and improved anonymity. Note that $r_1 \neq 0$

Note, that balance proof for such PoS miner transactions should take block reward into account, which is a trivial modification, because reward is public knowledge and nominated in native coins only.

4 Possible attack vector and mitigation

Asset tag concept is sensitive to cryptographic properties of the deterministic hash function H_p. Namely, we need to ensure that there's no efficiently-computable way to solve the following problem A:

Problem A: Let $H_t = H_p(T_0)$. Given H_t and T_0 find x, y such that:

$$H_p(x) = yH_p(T_0) = yH_t$$

Otherwise one would be able to generate arbitrary amount of assets while still keeping Eq. 5 hold.

The complexity of brute force attacks (including MITM) can possibly be increased by using computational-expensive hash function H_e to calculate asset-specific generator H_t:

$$H_t = H_p(H_e(asset\_\ descriptor\_\ t))$$

because normally the calculation of H_t is a rare operation and its result can be cached.

5 Efficiency

Let's try to estimate size of the all signatures and proofs for a transaction with m inputs (each having n elements in its ring) and k outputs.

Parameter	$\mathbb{G}$	$\mathbb{Z}_l$
Inputs (key images)	m	
Pseudo output commitments	$2m$	
Ring signature (CLSAG)		$(n+1)m$ [9]
Outputs' range proofs (BP+)	$k+\lceil(2\cdot\lceil\log_2(64)+\log_2(1)\rceil+3)\rceil$	2k + 4
Outputs' additional proofs		(km + 1) [10]
Outputs data (except proofs)	$3k$	k
Total	$3m+4k+15$	$(n+k+1)m+3k+5$

Table 1: Size comparison. $\mathbb{G}$ means the number of group elements, $\mathbb{Z}_p$ means the number of field elements.

References

[1] Jonathan Bootle and Jens Groth. *Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials*. Cryptology ePrint Archive, Paper 2018/045. https://eprint.iacr.org/2018/045. 2018. URL: https://eprint.iacr.org/2018/045.

[2] Jonathan Bootle et al. *Short Accountable Ring Signatures Based on DDH*. Cryptology ePrint Archive, Paper 2015/643. https://eprint.iacr.org/2015/643. 2015. URL: https://eprint.iacr.org/2015/643.

[3] Pyrros Chaidos and Vladislav Gelfer. *Lelantus-CLA*. https://eprint.iacr.org/2021/1036.pdf. 2021.

[4] Muhammed F. Esgin et al. *MatRiCT: Efficient, Scalable and Post-Quantum Blockchain Confidential Transactions Protocol*. https://eprint.iacr.org/2019/1287.pdf. 2019.

[5] Brandon Goodell, Sarang Noether, and Arthur Blue. *Concise Linkable Ring Signatures and Forgery Against Adversarial Keys*. https://eprint.iacr.org/2019/654.pdf. 2019.

[6] Jens Groth and Markulf Kohlweiss. *One-out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin*. Cryptology ePrint Archive, Paper 2014/764. https://eprint.iacr.org/2014/764. 2014. URL: https://eprint.iacr.org/2014/764.

[7] Gregory Maxwell. *Confidential Transactions*. https://web.archive.org/web/20200502151159/https://people.xiph.org/~greg/confidential_values.txt (Archived 2020-05-02). 2015.

[8] Shen Noether, Adam Mackenzie, and Monero Core Team. *Ring Confidential Transactions, MRL-0005*. https://web.getmonero.org/resources/research-lab/pubs/MRL-0005.pdf. 2016.

[9] Torben Pryds Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. https://www.cs.cornell.edu/courses/cs754/2001fa/129.PDF. 1992.

[10] Andrew Poelstra et al. *Confidential Assets*. 2018.

[11] Nicolas van Saberhagen. *CryptoNote v 2.0*. https://cryptonote.org/whitepaper.pdf. 2013.

[12] sowle. *d/v-CLSAG: Extension for Concise Linkable Spontaneous Anonymous Group Signatures*. https://hyle-team.github.io/docs/zano/dv-CLSAG-extension/dv-CLSAG-extension.pdf. 2024.

[13] sowle and koe. *Zarcanum: A Proof-of-Stake Scheme for Confidential Transactions with Hidden Amounts*. https://eprint.iacr.org/2021/1478.pdf. 2022.

[9] CLSAG compresses all additional layers.

[10] A ring signature for each output.