Zano: confidential assets scheme (DRAFT)

sowle

October 29, 2022

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.

1 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: $H_p : \{0,1\}^* \to \mathbb{G}$.

2 The concept

2.1 Asset descriptor

Our approach is based on the concept of confidential assets with blinded assets tags originally proposed in [6] and adapted in [3]. Here we briefly describe the concept.

In a normal confidential transaction each output's amount a is committed using Pedersen commitment scheme with additively homomorphic commitments:

$$A = aH + fG$$

where $H, G \in \mathbb{G}$ are group elements with no efficiently-computable discrete logarithm relation and $f \in \mathbb{Z}_l$ is a random blinding factor.

The initial emission of an asset in Zano will be made through a transaction which burns some amount of native coins and publicly discloses via the blockchain an *asset descriptor* with the following information associated with the given asset:

- unique name, symbol, text description, etc.;
- emission parameters (like total or unlimited supply);
- owner pubkey;
- etc.

The idea is to use a different amount-bonded generator $H_t = H_p(asset\_descriptor\_t)$ [1] for each asset t instead of the generator H. So the amount a for the asset t is committed to in:

$$A_t = aH_p(asset\_descriptor\_t) + fG$$

[1] In [6] such generators are called *asset tags*.

2.2 Hiding real asset tag

Consider slightly tweaked asset tag hiding scheme. Let $X \in \mathbb{G}$ be another generator, i.e. has no efficiently-computable discrete logarithm relation with both H and G. Then a typical transaction output would be represented by a tuple (hereinafter we omit output stealth address, encoded amount and other data for simplicity):

$$(\sigma^{rp}, T, A = aT + fG)$$

where $T = H_t + rX = H_p(asset\_descriptor\_t) + rX$ is blinded asset tag, $r = H_s(...)$ is a pseudo-random mask, known only to the output's owner and the sender, and σ^{rp} is a range proof for the fact, that $a < 2^{64}$.

Instead of disclosing asset tag H_t for blockchain observers verification, 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.

2.3 Balance proof

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

For example, consider a RingCT transaction with two inputs and four outputs ($m = 2, k = 4$). Each of input refers to a ring of three outputs ($n = 3$), and the second one is actually being spent.

#	Ring	Pseudo output commitments	Ring signature add. assertions	Outputs
0	$(\sigma_0^{rp}, T_0, A_0 = ?)$	$A_0^p = a_1 T_1 + f_1' G$	$A_0 - A_0^p = ?$ $T_0 - T_0^p = ?$	$E_0 = e_0(\underbrace{H_4 + r_0' X}) + y_0 G$ $(\sigma_0'^{rp}, T_0', E_0)$
	$(\sigma_1^{rp}, T_1, A_1 = a_1(\underbrace{H_1 + r_1 X}_{T_1}) + f_1 G)$	$T_0^p = T_1 + r_1' X$	$A_1 - A_0^p = (f_1 - f_1')G$ $T_1 - T_0^p = (r_1 - r_1')X$	$E_1 = e_1(\underbrace{H_4 + r_1' X}) + y_1 G$ $(\sigma_1'^{rp}, T_1', E_1)$
	$(\sigma_2^{rp}, T_2, A_2 = ?)$		$A_2 - A_0^p = ?$ $T_2 - T_0^p = ?$	$E_2 = e_2(\underbrace{H_1 + r_2' X}) + y_2 G$ $(\sigma_2'^{rp}, T_2', E_2)$
1	$(\sigma_3^{rp}, T_3, A_3 = ?)$	$A_1^p = a_4 T_4 + f_4' G$	$A_3 - A_1^p = ?$ $T_3 - T_1^p = ?$	$E_3 = e_3(\underbrace{H_4 + r_3' X}) + y_3 G$ $(\sigma_3'^{rp}, T_3', E_3)$
	$(\sigma_4^{rp}, T_4, A_4 = a_4(\underbrace{H_4 + r_4 X}_{T_4}) + f_4 G)$	$T_1^p = T_4 + r_4' X$	$A_4 - A_1^p = (f_4 - f_4')G$ $T_4 - T_1^p = (r_4 - r_4')X$	
	$(\sigma_5^{rp}, T_5, A_5 = ?)$		$A_5 - A_1^p = ?$ $T_5 - T_1^p = ?$	

In this example input 0 corresponds to the native coin, and so output E_2 (note using H_1). Hence $a_1 = e_2 + \mathit{fee}$[2]. Input 1 corresponds to an asset, associated with generator H_4, and outputs 0, 1, and 3 use the same asset-associated generator H_4. Hence, $a_4 = e_0 + e_1 + e_3$. Thanks to homomorphism we can combine both equations into one using corresponding commitments:

$$\sum_{i=0}^{m-1} A_i^p - \sum_{j=0}^{k-1} E_j - \text{fee} \cdot H_1 = rX + yG \tag{1}$$

where m is the number of inputs, and k is the number of outputs. Observers make sure that the equation above holds for some secret r, y using a vector Schnorr proof. Note, that r_j' and y_j are calculated using a shared secret ($r_j' = H_s(...), y_j = H_s(...)$) so output's receiver is able to calculate them.

2.4 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 [6], later improved in [3] with the help of logarithmic membership proof by Groth, Bootle at al. [5][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 $\{T_i^p - T_j'\}$:

$$T_i^p - T_j' = xX$$

[2]Here we assume that the transaction fee can only be paid in native tokens, and its plain unencoded value explicitly stated in transaction data.

(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).

2.5 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.

3 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:

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

Otherwise one would be able to generate arbitrary amount of assets while still keeping Eq. 1 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.

4 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)		$(n+1)m$ [3]
Outputs' range proofs (BP+)	$k+[(2\cdot\lceil\log_2(64)+\log_2(1)\rceil+3)]$	2k + 4
Outputs' additional proofs		(km + 1) [4]
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.

[3]CLSAG compresses all additional layers.

[4]A ring signature for each output.

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] 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.

[6] Andrew Poelstra et al. *Confidential Assets*. 2019.