3CMP: Indirect Index Integral Contents Mutation Protocol

Alisa Cherniaeva
=nil; Foundation
a.cherniaeva@nil.foundation

Ilia Shirobokov
=nil; Foundation
i.shirobokov@nil.foundation

Mikhail Komarov
=nil; Foundation
nemo@nil.foundation

July 2, 2021

1 Introduction

Authenticated cluster transaction log clusters preserve change history in explicit manner. Index changes are done with publicly auditable transactions which disables any kind of privacy.

Several cluster replication protocols provide stronger privacy guarantees [Noe15], [Mie+13], but since their index structure is unique, methods used by them don't suit for the incorporation into other protocols.

This paper proposes a more generic mechanism to provably hide the integral index contents changeset.

The proposed mechanism uses NIZK arguments for arithmetic circuit satisfiability and Elgamal cryptosystem [ElG85].

NIZK argument is instantiated as construction from [Gro16] but can be interchanged by other constructions to meet particular requirements.

2 Preliminaries

2.1 zk-SNARK

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) is a non-interactive zero knowledge argument of knowledge for NP that also achieves succinctness. ZK-SNARK contains three probalistic polynomial algorithms:

1. **Setup**(λ, C): for a given private parameter λ and an arithmetic circuit C, outputs public parameters pp.
2. **Prove**$(pp, pub\_in, priv\_in, out)$: generates an argument π that $C(pub\_in, priv\_in) = out$ without disclosure any information about $priv\_in$
3. **Verify**(pp, pub_in, out, π): checks that π is a valid argument for (C, pub_in, out, pp)

We are focused on the R1CS-based zk-SNARKs. In particular, we use a Groth[Gro16] construction because of the verifier efficiency. However, the proposed system can be used with other zk-system constructions (e.g. [Par+13], [Mal+19], [GWC19]), including transparent constructions that do not require trusted setup (e.g. [Set19], [BFS19]).

2.2 Elgamal Cryptosystem

Elgamal encryption is described by Taher Elgamal in [ElG85]. In this work, we use elliptic curve version of the cryptosystem. Let p be a large prime number. The elliptic curve E is defined over a field $\mathbb{F}_p$, its cyclic subgroup $\mathbb{G} \in E(\mathbb{F}_p)$ has prime order q. Generator of group $\mathbb{G}$ is G.

Elgamal Cryptosystem defined by the following algorithms:

- **Setup**(1^λ): Choose a private key $x \xleftarrow{\text{R}} \mathbb{Z}_q$ and a public key $Q = xG$.
- **Encrypt**(Q, m): Map message m to the point $M \in \mathbb{G}$. The ciphertext is $C = (rG, M + rQ)$, $r \xleftarrow{\text{R}} \mathbb{Z}_q$.
- **Decrypt**(C, x): For the ciphertext $C = (A, B) \in E(\mathbb{F}_p) \times E(\mathbb{F}_p)$, decrypted point is $M = B - xA$. Decode point M to the message m.

The mapping into a point on the elliptic curve has to have an efficiently calculated reverse mapping function for big enough message space. The probabilistic embedding from N. Koblitz[Kob87] can be used for these purposes.

Let m be the message to be mapped to an elliptic curve. We choose a random integer r and concatenate $(r||m) < q$. It would be a potential x-coordinate of point M. If the solution of a quadratic equation of elliptic curve exists then we found the point M. Else we try with a new random integer r. Thus, the reverse mapping is a simple dropping of the bits that relate to r from the x-coordinate. This is a probabilistic embedding so there is a negligible probability that the point M does not exist. If $p = 3 \mod 4$, then r takes about 10 bits.

3 Proposal

The protocol contains two parts:

1. **Circuit Components**. Required for proof generation.
2. **Participant's Communication Protocol**. Required to define participants' behavior.

3.1 Circuit Components

In this section, basic components of our ZK-SNARK circuit get defined.

3.1.1 Range Check Circuit's Component

The range circuit's component checks that the value lies in a certain interval.

Suppose that the prover wants to convince the verifier that the witness w is less than c. Let n be a public value, such that $c < 2^n$.

Algorithm 1 Range Check Circuit (RCC)

Public Input: n
Private Input: $w, c, \alpha, \{\alpha_i \text{ for } i = 0, ..., n\}, d$

1. $(2^n + w - c) \times (1) = (\alpha)$.
2. $(\sum_{i=0}^{n} \alpha_i 2^i) \times (1) = \alpha$
3. $(1 - \alpha_i) \times (\alpha) = (0)$ for $i = 0, ..., n$
4. $(\sum_{i=0}^{n} \alpha_i) \times (1 - d) = (0)$, where $d = \bigvee_{i=0}^{n-1} \alpha_i$
5. $(\sum_{i=0}^{n} \alpha_i) \times (\sum_{i=0}^{n} \alpha_i)^{-1} = (d)$
6. $(d) \times (1 - d) = (0)$
7. $(\alpha_n) \times (d) = (0)$

Note, that the value c can be public as well.

3.1.2 Elgamal Encryption Check Circuit's Component

In order to prove the correctness of ElGamal encryption, the following functions are presented. The basic operations on the elliptic curve in the ZK-SNARK circuit are out of the scope of this paper. For this reason, the following protocols use the elliptic curve's arithmetic as a black box.

Proof of Encryption Let Prover has a public key Q of the verifier, but knows nothing about the secret key. They proves a knowledge of the random integer r and the message m in the encrypted message (A, B).

Algorithm 2 Encryption Check Circuit (ECC)

Public Input: $G, Q, (A, B)$
Private Input: r, m, M

1. $rG - A == \mathcal{O}$
2. $rQ + M - B == \mathcal{O}$
3. Check that $M = (X_m, Y_m)$ lies on the elliptic curve
4. $(\sum_{i=0}^{N} m_i 2^i) \times (1) = (X_m)$
5. $(1 - m_i) \times (m_i) = (0)$ for $i = 0, ..., N$
6. $(\sum_{i=0}^{l} m_i 2^i) \times (1) = (m)$

The steps 1 and 2 relate to Elgamal encryption. These steps allow to convince the verifier, that the prover knows the random r and the point M used in the private input. The remaining steps prove that m contains in the x-coordinate of M, which have to belong to the defined elliptic curve. The step 4 checks that the sequence of bits $m_i, i = 0, .., N$ fits to X_M and the step 5 checks that $m_i \in \mathbb{B}$. Clearly, the step 6 means that the sequence of bits $m_i, i = 0, .., l$ fits to m.

Proof of Decryption Decryption Check Circuit acts similarly to Encryption Check Circuit except the part with knowledge of the secret key.

Algorithm 3 Decryption Check Circuit (DCC)

Public Input: $G, Q, (A, B)$
Private Input: x, m, M

1. $xG - Q == \mathcal{O}$
2. $B - xA - M == \mathcal{O}$
3. Check that $M = (X_m, Y_m)$ on the elliptic curve
4. $(\sum_{i=0}^{N} m_i 2^i) \times (1) = (X_m)$
5. $(1 - m_i) \times (m_i) = (0)$ for $i = 0, ..., N$
6. $(\sum_{i=0}^{l} m_i 2^i) \times (1) = (m)$

3.1.3 Integral Index Changeset Circuit Component

For requirements of confidentiality, it is required to hide each integral index value and the index integral value difference within the changeset.

Let the prover with an integral index value a (which is encrypted as $A = (A_1, A_2)$) wants to introduce an integral value changeset $z \leq a$ to the verifier's index. The new Prover's integral value $a' = a - z$ is encrypted as $A' = (A'_1, A'_2)$.

Assume that the prover's pair of private and public keys is (x_P, Q_P) and verifier's public key is Q_V. The zk-SNARKs circuit in the Prover's phase proceeds as in (Algorithm 4).

Algorithm 4 Integral Index Changeset Circuit

Public Input $A = (A_1, A_2), B = (B_1, B_2), A' = (A'_1, A'_2), H, Q_P, Q_V$
Private Input $x_P, a, z, a', r_1, r_2, M_a, M_z, M_{a'}$

1. **DCC**$(H, Q_P, (A_1, A_2), x_P, a, M_a)$, where M_a is encoded a
2. **ECC**$(H, Q_V, (B_1, B_2), r_1, z, M_z)$, where M_z is encoded z
3. The Prover creates the **RCC** for $z \leq a$.
4. $a' = a - z$
5. **ECC**$(H, Q_P, (A'_1, A'_2), r_2, a', M_{a'})$, where $M_{a'}$ is encoded a'

Proof. Firstly, it is required to prove that the **RCC** for $z \leq a$ shows that Prover's index integral value will not break the local storage consistency (the chageset index integral value difference is positive).

Since $2^n + z - a \leq 2^n \Leftrightarrow z \leq a$, the coefficient α_n is 1 in the case, when $z = a$, and 0 otherwise. The value $d = \bigvee_{i=0}^{n-1} \alpha_i$ equals 0, if $z = a$ and 1 otherwise. The checks that $\alpha_i, d \in \mathbb{B}$ are added explicitly. As a result, the final step $\alpha_n \cdot d = 0$ can hold iff $z \leq a$.

Secondly, the prover's index integral value a and an the changeset value difference z are encrypted so it is required to map the values from the statement above to encrypted values and vice versa. The algorithms **ECC** and **DCC** repeat steps from the original algorithms Elgamal Encryption Scheme and Koblitz's probabilistic embedding and save each step as part of the circuit. The correctness of the circuit obviously follows from that fact.

Now consider the case, when the changeset integral index value $\hat{z}$ is negative. Since the field $\mathbb{F}_q$ is used, then $\hat{z} = q - \hat{z}$ is used in real calculations. Therefore, if $q - \hat{z} \leq a$ then the proof and

the changeset are valid. The value $a' = a - (q - \hat{z}) \geq 0$ is correct. If the statement $q - \hat{z} \leq a$ is not held then the proof is not valid.

Since the homomorphic encryption is not being used, the expression $a' = a - z$ checks a'. Then, its encryption gets proved in the same manner as z. This approach allows the prover to convince the verifier that the prover's integral index value and its changeset value are consistent. □

Third Party Verifier In case a third-party has to be able to verify that the Prover's index integral value is consistent, but cannot decrypt the ciphertext, the proving of statement $z \leq a$ is not enough.

If the expected index changeset integral value is a public value z, then an additional step of the protocol is added as follows:

Algorithm 5 Integral Index Changeset Circuit (Public Integral Value)

Public Input $A = (A_1, A_2), B = (B_1, B_2), A' = (A'_1, A'_2), H, Q_P, Q_V$
Private Input $x_P, a, z, a', r_1, r_2, M_a, M_z, M_{a'}$

1. **DCC**$(H, Q_P, (A_1, A_2), x_P, a, M_a)$, where M_a is encoded a
2. **ECC**$(H, Q_V, (B_1, B_2), r_1, z, M_z)$, where M_z is encoded z
3. The Prover creates the **RCC** for $z \leq a$.
4. $a' = a - z$
5. **ECC**$(H, Q_P, (A'_1, A'_2), r_2, a', M_{a'})$, where $M_{a'}$ is encoded a'
6. The Prover creates the **RCC** for $z \leq z'$, where z' is an index changeset integral value.

Since **RCC** does not require many constraints, the total size of the circuit does not increase significantly.

If z is stored as an encrypted value $Enc(z, Q_P) = (Z_1, Z_2)$, then two additional steps are added as follows:

Algorithm 6 Integral Index Changeset Circuit (Encrypted Integral Value)

Public Input $A = (A_1, A_2), B = (B_1, B_2), A' = (A'_1, A'_2), H, Q_P, Q_V$
Private Input $x_P, a, z, a', r_1, r_2, M_a, M_z, M_{a'}$

1. **DCC**$(H, Q_P, (A_1, A_2), x_P, a, M_a)$, where M_a is encoded a
2. **ECC**$(H, Q_V, (B_1, B_2), r_1, z, M_z)$, where M_z is encoded z
3. The Prover creates the **RCC** for $z \leq a$.
4. $a' = a - z$
5. **ECC**$(H, Q_P, (A'_1, A'_2), r_2, a', M_{a'})$, where $M_{a'}$ is encoded a'
6. **DCC**$(H, Q_P, (Z_1, Z_2), x_P, z, M_z)$, where M_z is encoded z.
7. The Prover creates the **RCC** for $z \leq z'$, where z' is an index changeset integral value.

3.2 Communication Protocol

The protocol proposed containts two kinds of indexes:

1. **Participant's Index**. Contains mutable relations mapped to participant's unique identifier. Accepts mutations from participant only. Such relations may contain: integral values i_n mapped to participant n, various notes.
2. **Root Index**. It is supposed to store non-mutable protocol parameters such as: participants amount N, index integral values maximum $\sum_{n=0}^{N-1} i_n \equiv C \ \forall C \in \mathbb{Z}$ or any other params. This index has it's participant's unique identifier mapping as well (aka "index owner").

3.2.1 Confidential Participant's Index Integral Changeset Protocol

Considering the communication protocol, let participant A wants to introduce some index integral value changeset z to an index which already has value $a : a \geq z$ mapped to participant A and some value mapped to some participant B. The key pair of A is Q_A, x_A, and Q_B is a public key of the participant B. The protocol supposes every step is being performed over consistent data storage.

Algorithm 7 Confidential Index Changeset

1. The participant A decrypts $a = Dec(x_A, (A_1, A_2))$.
2. The participant A encrypts $Z = Enc(z, Q_B) = (r_1 G, r_1 Q_B + M_z) = (Z_1, Z_2)$ for $r_1 \xleftarrow{\mathrm{R}} \mathbb{F}_q$.
3. The participant A encrypts $R = Enc(a - z, Q_A) = (r_2 G, r_2 Q_A + M_{a-z}) = (R_1, R_2)$ for $r_2 \xleftarrow{\mathrm{R}} \mathbb{F}_q$.
4. The participant A generates proof π_{AB}, that $z \leq a$ as in 4.
5. The participant A sends Z, R and the proof π_{AB} to B.
6. The participant's index integral value A sets on R.

Proof Sketch. The proof idea is that the original protocol could be reduced (using framework [Sho04]) to calculation through a trusted third party (TTP) where both participants cannot influence it and cannot get any information about secret values. Here the main steps of the reduction are mentioned.

From the security of the SNARK protocol, it follows that the arithmetic circuit can be calculated by a TTP with data sent from the prover via a private channel. The circuit's correctness is proved above. The security properties of Elgamal encryption allow hiding participants integral index values and the index changeset value difference on the TTP side and show random ciphertexts instead of the real one. Thus, all sensitive calculations can be brought to the TTP side. □

Index integral values aggregation The changeset integral value (or a wrapping relation) is being written-in to B's integral value index as a new entity. In order to send few entities, the index owner adds **DCC** for each additional entity into zk-SNARK. For the requirements of efficiency, we bound the amount of these entities by L. Additionally, the participant can send an aggregation transaction which is also bounded by L.

Let the index owner has $n < L$ entities e_i. The aggregation transaction is created as follows:

Algorithm 8 Changeset Aggregation Circuit

1. The index owner creates the **DCC** for each e_i.
2. $E = \sum_{i=0}^{n} e_i$
3. The index owner creates the **ECC** for E.

3.2.2 Confidential Root Index Integral Changeset Protocol

Confidential root index integral values changeset now gets described. This case supposes there is no need to prove the after-changeset value, because the root index relations do not store it. Instead of this, the index owner has to prove the new granted index integral value g'.

Let a root index has N integral value written-in, where N is a public value (aka *totalSupply*). The total index integral value is a value $g = \sum_{n=0}^{N-1} i_n$ or *totalGranted*, which is stored as an Elgamal ciphertext $G = (G_1, G_2)$. The key pair of an index owner is Q_P, x_P.

Algorithm 9 Root Index Changeset

1. The index owner calculates some unique address of the participant's index.
2. The index owner encrypts the integral value from the new index b as $B = Enc(b, Q_V) = (r_b H, r_b Q_V + M_b) = (B_1, B_2)$ for $r_b \xleftarrow{R} \mathbb{F}_q$ using the public key Q_V of this index.
3. The index owner decrypts $g = Dec(x_P, G)$.
4. The index owner increases the integral value $g' = g + b$ and encrypts as $G' = Enc(g', Q_P) = (r'_g H, r'_g Q_P + M'_g) = (G'_1, G'_2)$ for $r'_g \xleftarrow{R} \mathbb{F}_q$.).
5. The index owner generates proof, that the sum of b and g is less or equal than N:
   - 5.1 $\mathbf{ECC}(H, Q_V, (B_1, B_2), r_b, b, M_b)$, where M_b is encoded b
   - 5.2 $\mathbf{DCC}(H, Q_P, (G_1, G_2), x_P, g, M_g)$, where M_g is encoded g
   - 5.3 The index owner creates the **RCC** for the $b + g \leq N$.
   - 5.4 $\mathbf{ECC}(H, Q_P, (G'_1, G'_2), r'_g, g', M'_g)$, where M'_g is encoded g'
   - 5.5 $g' = g + b$.
6. The index owner asks the root index to emplace the data defined within the participant's index.
7. The index owner updates the *totalGranted* value.

Proof Sketch. The validity of Root Index Changeset circuit index integral contents follows from the integral index changeset circuit correctness proof 4 and the root index interaction part gets reduced to TTP computations using framework [Sho04] just as it is done with 3.2.1. □

Remark. The root index does not hide the changeset integral value, because the *totalSupply* is a public value. The details about changing such immutable root index parameters are out of the scope of this paper since it is a matter of a particular index management logic implementation.

3.3 Participants Identifiers Confidentiality Extension

Protocol extension covering hiding participant's index identifiers is also possible according to the scheme described. The difference would be about circuit components's data put under the ElGamal scheme extended from having only index integral value g to the tuple (Q_V, g) containing participants' public identifiers as well.

References

[BFS19] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. *Transparent SNARKs from DARK Compilers*. Cryptology ePrint Archive, Report 2019/1229. https://eprint.iacr.org/2019/1229. 2019.

[ElG85] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472.

[Gro16] Jens Groth. "On the size of pairing-based non-interactive arguments". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2016, pp. 305–326.

[GWC19] Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge." In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 953.

[Kob87] Neal Koblitz. "Elliptic curve cryptosystems". In: *Mathematics of computation* 48.177 (1987), pp. 203–209.

[Mal+19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. "Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2111–2128.

[Mie+13] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. "Zerocoin: Anonymous distributed e-cash from bitcoin". In: *2013 IEEE Symposium on Security and Privacy*. IEEE. 2013, pp. 397–411.

[Noe15] Shen Noether. "Ring SIgnature Confidential Transactions for Monero." In: *IACR Cryptol. ePrint Arch.* 2015 (2015), p. 1098.

[Par+13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. "Pinocchio: Nearly practical verifiable computation". In: *2013 IEEE Symposium on Security and Privacy*. IEEE. 2013, pp. 238–252.

[Set19] Srinath Setty. *Spartan: Efficient and general-purpose zkSNARKs without trusted setup*. Cryptology ePrint Archive, Report 2019/550. https://eprint.iacr.org/2019/550. 2019.

[Sho04] Victor Shoup. "Sequences of games: a tool for taming complexity in security proofs." In: *IACR Cryptol. ePrint Arch.* 2004 (2004), p. 332.

Appendices

A Telegram Open Network Terms

Telegram Open Network calls integral index contents mapped to some unique identifier a "Wallet". Integral value (or a wrapping relation) itself is called a "Token". There are several types of wallets:

- Root Token Wallet - a merge operator that deploys a TON Token wallets. It supports the only one token type. The **token owner** deploys the root token wallet with a root public key as a part of initial data. It contains the total number of tokens N and the number of the granted tokens g.
- TON Token Wallet - a merge operator deployed by root wallet. One **token wallet owner** can have several wallets. Anyone who has the TON token wallet code, the root public key, and the root wallet address can calculate the TON wallet address and deploy the wallet. When the wallet was constructed the *StateInit* structure from the inbound message to store it in its persistent storage for later use.

B Telegram Open Network UTXO Use Case

The participant A has UTXO-based wallet with a tokens, where Q_A, x_A are public and private keys. They send $z, z \leq a$ tokens to participant B, which public key is Q_B, as follows:

Algorithm 10 UTXO Confidential Transaction

1. The participant A decrypts $a = Dec(x_A, (A_1, A_2))$.
2. The participant A encrypts $Z = Enc(z, Q_B) = (r_1 G, r_1 Q_B + M_1) = (A_1, B_1)$ for $r_1 \xleftarrow{\text{R}} \mathbb{F}_q$.
3. The participant A encrypts $R = Enc(a - z, Q_A) = (r_2 G, r_2 Q_A + M_2) = (A_2, B_2)$ for $r_2 \xleftarrow{\text{R}} \mathbb{F}_q$.
4. The participant A generates proof π_{AB}, that $z \leq a$.
5. The participant A calculates an address of new wallet using the B's public key.
6. The participant A deploys a new wallet at the address from the previous step.
7. The participant A sends Z, R, π_{AB} to new wallet of the partiicipant B.
8. The participant A calculates an address of the new wallet with own public key.
9. The participant A deploys a new wallet at the address from the previous step.
10. Th participant A sends Z, R, π_{AB} to new wallet of the partiicipant A.

These actions perform as one transaction.

The deployment of UTXO-based wallet proceeds as for Token wallets. However, the initial data of a UTXO wallet contains an additional `utxoFlag` boolean flag set to false. When a wallet receives tokens, it is switched to true. Note, that the zero tokens in the transaction are not accepted.

The aggregation proceeds as described for the token wallets, but instead of several balances on the one wallet, the aggregation of several wallets is used.