Advancing Blockchain Transaction Privacy and Compliance: Insights into Innovative Engineering Practices

Authors: Lone(ZKT), Taiyo(ZKT), Xie. X(PADO Labs)

1. Introduction

Blockchain privacy and regulatory compliance are experiencing a dynamic shift, significantly inspired by the pivotal works of Vitalik Buterin and Ameen Soleimani. Their insightful paper, "Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium[1]," along with the enlightening forum post on permissioned privacy pools, has laid a foundational framework for understanding the intricate balance between maintaining transactional privacy and adhering to regulatory norms. These resources delve deeply into the challenges and viable solutions for synchronizing privacy with compliance in the ever-evolving blockchain landscape.

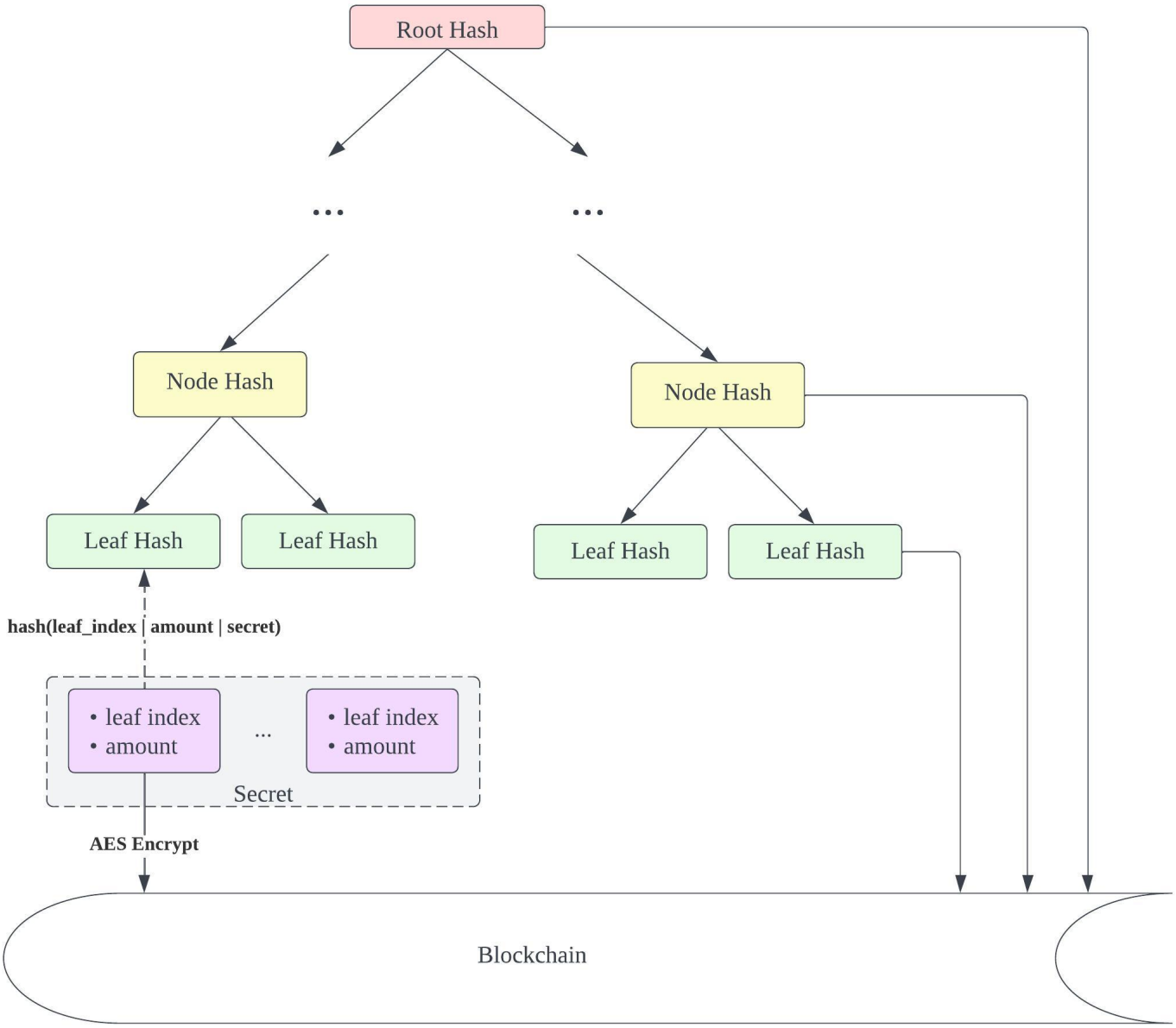
A key takeaway from their works is the concept of 'honest address sets' which profoundly influences our approach to achieving a second-generation privacy protocol. In this framework, Association Set Providers (ASPs) emerge as crucial facilitators of credit within the zk-credit system, offering a structured approach to assess and attest to the creditworthiness of users. Meanwhile, membership proof ensures that all participants in a transaction are part of an honest address set, enhancing the security of our protocol and providing a safer, more trustworthy environment for users.

In summary, the works of Buterin and Soleimani have been instrumental in guiding our approach to developing a blockchain system that upholds both privacy and regulatory compliance. By integrating these concepts into our system, we have been able to create a more secure, efficient, and privacy-preserving second-generation protocol.

2. System Architecture

At the forefront of this evolution is Vala, an innovative settlement layer that embodies the principles of compliant privacy transactions. Vala integrates a multi-chain cross-chain module, fostering seamless interoperability across various blockchain networks. This feature is pivotal in enabling a boundless crypto ecosystem, where asset transfer and management are not confined to the boundaries of a single blockchain.

Central to Vala's architecture is the Privacy Pool, a sophisticated mechanism that allows users to engage in UTXO (Unspent Transaction Output) model transactions. This model is instrumental in enhancing user privacy while ensuring transactions remain transparent and compliant with regulatory standards. The UTXO model, a hallmark of blockchain technology, offers an efficient way to track asset ownership and transfer, laying the groundwork for secure and private transactions within the Vala ecosystem.



3. Technology Stack

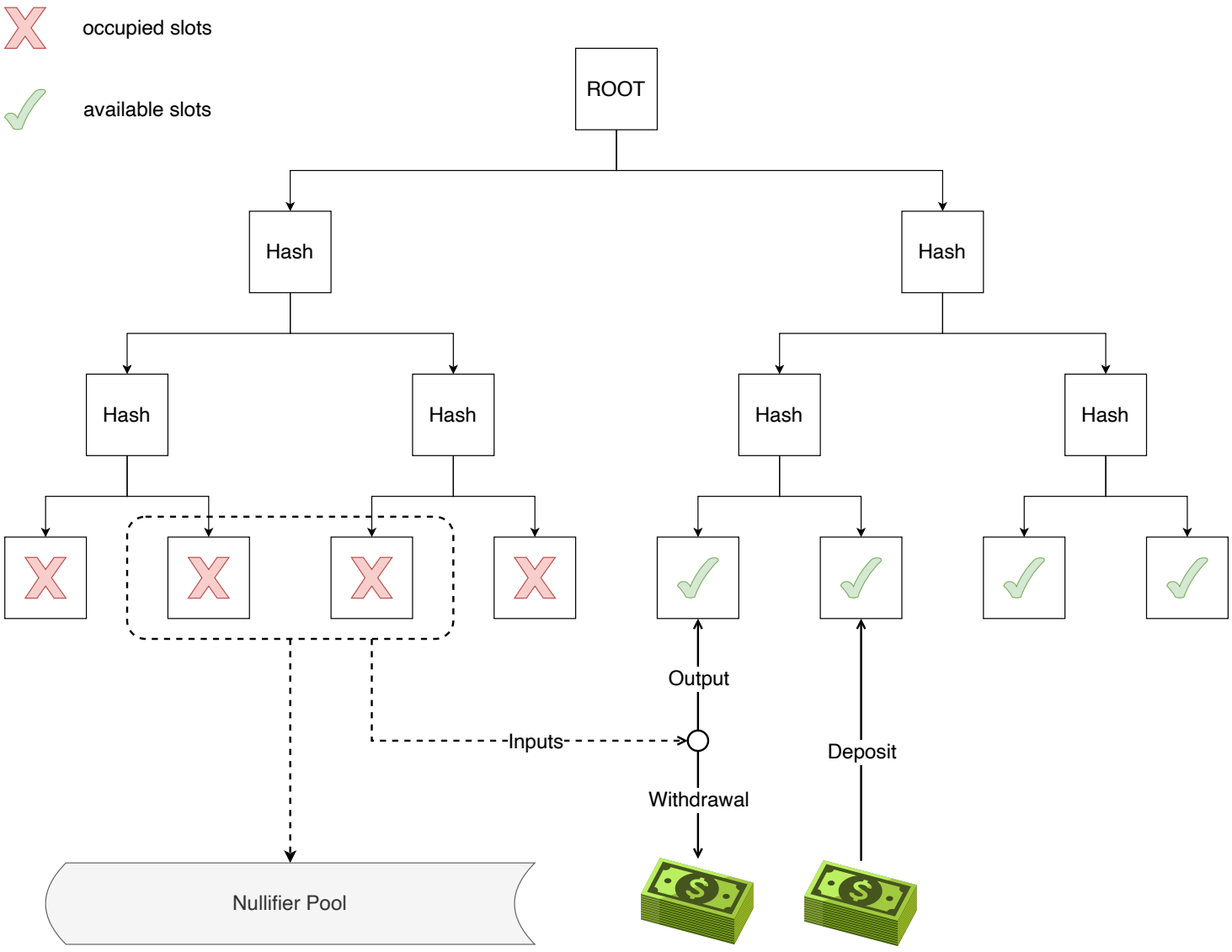
The technology stack for ZKT Vala is anchored by zkSNARK for privacy preservation. The UTXO Proof uses a Merkle Tree structure, while Membership Proof employs Plonk + Plookup[2] for enhanced search speed. The project's zero-knowledge proofs part is implemented in Rust, optimizing performance and security. For the frontend, proof generation utilizes WebAssembly compiled from our Rust code, significantly accelerating the proof generation process for users. This stack represents a blend of advanced cryptographic techniques and efficient programming solutions, tailored for robust and fast privacy transactions.

4. Component Details

4.1 Infrastructure

The users' fungible assets are hashed into a Binary Merkle Tree. The leaf node is initialized with a constant value $H' = \text{Hash}(0)$ until any asset occupies the slot of the leaf, then the leaf node should be updated to $H = \text{Hash}(identifier \mid amount \mid commitment)$, where $identifier$ is an associated tag of this asset (like the user's address), $amount$ is the quantity of the asset stored at the leaf node, and $commitment = \text{Hash}(secret)$, with $secret$ being the key of the user who owns the asset.

For an unused leaf node slot, the user can deposit an asset into the pool and take the slot while providing the asset and commitment. For withdrawal, the user should provide the withdrawal amount, new leaf hash, and the snark proof.



4.2 Proof of ULO (Unspent Leaf Output)

The design of the withdrawal mechanism is influenced by the Bitcoin UTXO (Unspent Transaction Output) model, which we've adapted into what we call ULO (Unspent Leaf Output). Users select a set of leaf nodes that indicate

their assets to use as inputs, then hash the remaining balance into a new leaf node as output. The difference between the total input and the output is the amount of the withdrawal.

Now let the ULO source list be L, for each ULO with an index $i \in L$, we must first verify its existence and then calculate the corresponding leaf's nullifier within the circuit as follows:

$$\begin{aligned} commitment_i &= \mathrm{Hash}(secret_i) \\ nullifier_i &= \mathrm{Hash}(secret_i^{-1}) \\ leaf_i &= \mathrm{Hash}(identifier_i \mid amount_i \mid commitment_i) \\ root &= \mathrm{MerklePath}(leaf_i,\ elements) \end{aligned}$$

Next, we need to demonstrate that the total input amount is equal to the total output amount within the circuit, and then generate the leaf output. The following equations represent this:

$$\begin{aligned} \sum_{i \in L} amount_i &= amount_w + amount_o \\ commitment &= \mathrm{Hash}(secret) \\ leaf &= \mathrm{Hash}(identifier \mid amount_o \mid commitment) \end{aligned}$$

Where $amount_w$ is the withdrawal amount, $amount_o$ is the output amount.

Note
In the above process, we set $\{nullifier_i\}_{i \in L}$, $leaf$, $root$ and $amount_w$ as public variables from the circuits.

4.3 Proof of membership

4.3.1 Why not Merkle proof

Proof of membership in Vala lies in demonstrating that the inputs come from an arbitrary set constructed by the user, meanwhile ensuring that this set is publicly available to anyone. Typically, the set can be organized into a new Merkle Tree and the user should prove that each input is also a leaf node of the Merkle Tree (Indeed, privacy-focused solutions like Tornado Cash v2, and Privacy Pools have adopted this kind of design).

Apparently, if the set is too small, the user's privacy is compromised. If the set is too large, generating the corresponding Merkle tree incurs a huge gas cost on EVM, since ZK-friendly hash functions (Poseidon, Rescue) are not integrated by EVM primitively while creating Merkle tree has a complexity of O(n) hash.

We hereby adopt the Plookup to construct proof of membership, instead of Merkle Tree. Let’s take a look at the design idea of Plookup:

"*we precompute a lookup table of the legitimate (input, output) combinations; and the prover argues the witness values exist in this table.*"

There are **2** main advantages of Plookup in our membership proof.

- In the proving phase, we do not need to perform existence proofs for each ULO, which significantly reduces the circuit size.
- In the verification phase, we do not need to perform hash operations in EVM. Only one elliptic curve pairing is required.

4.3.2 Advantages of Plookup

Constant Complexity in Circuit Size

Traditionally, Merkle proof methods, when handling a large number of ULO proofs, lead to linear growth in circuit size due to the need for individual existence proofs for each ULO. This results in increased computational complexity and time costs. However, a key advantage of using Plookup in our UTXO model is that it maintains a constant level of complexity in circuit size. Regardless of the number of ULOs to be verified, Plookup can process them within a constant circuit size, significantly reducing the overall circuit size and processing time.

Efficiency in the Verification Process

Another advantage of Plookup in the verification phase is the elimination of the need for extensive hash operations in the Ethereum Virtual Machine (EVM). In contrast, traditional Merkle proofs require hashing at each node during verification, which is not only time-consuming but also computationally costly. Plookup simplifies the verification process by using a single elliptic curve pairing operation to ascertain the existence of ULOs. This approach not only speeds up the verification but is also more efficient in conserving computational resources.

Application of Plookup in the UTXO Model

In our UTXO model, each user's funds are represented as multiple encrypted ULOs, each indicating a specific amount of assets. With Plookup, we can effectively process and verify these ULOs irrespective of their quantity. This method enhances the scalability and efficiency of the entire system, particularly in handling a large volume of transactions and complex fund movements.

In summary, Plookup provides significant benefits in our membership proof system, especially in managing numerous ULOs within the UTXO model. This approach not only reduces computational burden but also enhances the overall system efficiency and performance.

4.3.3 Plookup introduce

Assuming the user provides an identifier list of size m from the source ULOs, denoted as $\mathbf{t}$, then we pad the set $\mathbf{t}$ with 0 until it satisfies the size of circuit size n.

$$\mathbf{t} = \{\mathrm{id}_0, \mathrm{id}_1, ..., 0, ..., 0\}$$

Let $\mathbf{f}$ be the query table:

$$f_i = q_{Ki} \cdot c_i = \begin{cases} c_i, & \text{if the } i \text{ gate is a lookup gate} \\ 0, & \text{otherwise} \end{cases}$$

Where c_i is the output witness defined in arithmetic gate of Plonk, we activate it when c_i represents the identifier witness. $q_{Ki} \in \{0, 1\}$ is the selector that switches on/off the lookup gate.

Furthermore, we define $\mathbf{q_T}$ to satisfy $q_{Ti} \cdot t_i = 0$:

$$q_{Ti} = \begin{cases} 0, & i \leq m \\ 1, & i > m \end{cases}$$

During the verification phase, besides the zk-SNARK proof verification, we can verify the opening proof of $\mathbf{t}$ at $\{1, \omega, \omega^2, ..., \omega^{m-1}\}$ to confirm the correctness of each identifier in $\mathbf{t}$, without any hash operations.

Note:
We can construct an aggregated opening proof, which is called a multi-point opening, and use only one elliptic curve pairing operation.

5. ZKT Protocol

We now describe a protocol that referred to Plonkup[3] which is customized for ZKT Vala.

Common Referenced Input

$$\begin{gathered} n,\ [1]_1, [x]_1, ..., [x^{n+5}]_1, [1]_2, [x]_2, \\ q_L(X) = \sum_{i=1}^{n} q_{Li} L_i(X),\ q_R(X) = \sum_{i=1}^{n} q_{Ri} L_i(X),\ q_O(X) = \sum_{i=1}^{n} q_{Li} L_i(X), \\ q_C(X) = \sum_{i=1}^{n} q_{Ci} L_i(X),\ q_K(X) = \sum_{i=1}^{n} q_{Ki} L_i(X),\ q_T(X) = \sum_{i=m+1}^{n} L_i(X), \\ S_{\sigma 1}(X) = \sum_{i=1}^{n} \sigma_{1i} L_i(X),\ S_{\sigma 2}(X) = \sum_{i=1}^{n} \sigma_{2i} L_i(X),\ S_{\sigma 3}(X) = \sum_{i=1}^{n} \sigma_{3i} L_i(X) \end{gathered}$$

Public input

$$\mathrm{PI}(X) = \sum_{i=1}^{l} w_i L_i(X)$$

5.2 Proving Process

Round 1

- Generate randome blinding scalars $b_1, ..., b_6$.
- Compute the wire polynomials $a(X), b(X), c(X) \in \mathbb{F}_{<n+2}[x]$:

$$a(X) = (b_1(X) + b_2) Z_H(X) + \sum_{i=1}^{n} w_{Li} L_i(X)$$

$$b(X) = (b_3(X) + b_4) Z_H(X) + \sum_{i=1}^{n} w_{Ri} L_i(X)$$

$$c(X) = (b_5(X) + b_6) Z_H(X) + \sum_{i=1}^{n} w_{Oi} L_i(X)$$

- Compute $[a(x)]_1$, $[b(x)]_1$, $[c(x)]_1$. Append them into `transcript`.

Round 2

- Compute the table polynomial $t(X) \in \mathbb{F}_m[x]$:

$$t(X) = \sum_{i=1}^{m} t_i L_i(X)$$

- Let $\mathbf{s} \in \mathbb{F}^{2n}$ be the vector that is $(\mathbf{f}, \mathbf{t})$ sorted by $\mathbf{t}$. We represents s by the vectors $\mathbf{h_1}, \mathbf{h_2} \in \mathbb{F}^n$ as follows:

$$\mathbf{h_1} = (s_1, s_3, ..., s_{2n-1})$$
$$\mathbf{h_2} = (s_2, s_4, ..., s_{2n})$$

- Generate random blinding scalars $b_7, ..., b_{11}$.
- Compute the polynomials $h_1(X) \in \mathbb{F}_{<n+3}[x]$, and $h_2(X) \in \mathbb{F}_{<n+2}[x]$.

$$h_1(X) = (b_7 X^2 + b_8 X + b_9) Z_H(X) + \sum_{i=1}^{n} s_{2i-1} L_i(X)$$

$$h_2(X) = (b_{10} X^2 + b_{11} X) Z_H(X) + \sum_{i=1}^{n} s_{2i} L_i(X)$$

- Compute $[t(x)]_1$, $[h_1(x)]_1$, $[h_2(x)]_1$. Append them into `transcript`.

Round 3

- Compute the permutation challenges $\beta, \gamma, \delta, \varepsilon \in \mathbb{F}$:

$$\beta = \text{Hash}(\text{transcript} \mid 1)$$
$$\gamma = \text{Hash}(\text{transcript} \mid 2)$$
$$\delta = \text{Hash}(\text{transcript} \mid 3)$$
$$\varepsilon = \text{Hash}(\text{transcript} \mid 4)$$

- Generate random blinding scalars $b_{12}, ..., b_{17}$.
- Compute the Plonk permutation polynomial $z_1(X) \in \mathbb{F}_{<n+3}[x]$:

$$z_1(X) = (b_{12} X^2 + b_{13} X + b_{14}) Z_H(X) + L_1(X) + \sum_{i=1}^{n-1} \left(L_{i+1}(X) \prod_{j=1}^{i} \frac{(w_{Lj} + \beta \omega^{j-1} + \gamma)(w_{Rj} + \beta k_1 \omega^{j-1} + \gamma)(w_{Oj} + \beta k_2 \omega^{j-1} + \gamma)}{(w_{Lj} + \beta \sigma_{1j} + \gamma)(w_{Rj} + \beta \sigma_{2j} + \gamma)(w_{Oj} + \beta \sigma_{3j} + \gamma)} \right)$$

- Compute the plookup permutation polynomial $z_2(X) \in \mathbb{F}_{<n+3}[x]$:

$$z_2(X) = (b_{15} X^2 + b_{16} X + b_{17}) Z_H(X) + L_1(X) + \sum_{i=1}^{n-1} \left(L_{i+1}(X) \prod_{j=1}^{i} \frac{(1+\delta)(\varepsilon + q_{Kj} c_j)(\varepsilon(1+\delta) + t_j + \delta t_{j+1})}{(\varepsilon(1+\delta) + s_{2j-1} + \delta s_{2j})(\varepsilon(1+\delta) + s_{2j} + \delta s_{2j+1})} \right)$$

- Compute $[z_1(x)]_1$ and $[z_2(x)]_1$.

Round 4

- Compute the quotient challenge $\alpha \in \mathbb{F}$:

$$\alpha = \text{Hash}(\text{transcript})$$

- Compute the quotient polynomial $q(X) \in \mathbb{F}_{<3n+5}[x]$:

$$q(X) = \frac{1}{Z_H(X)} \begin{pmatrix} a(X)b(X)q_M(X) + q(X)q_L(X) + b(X)q_R(X) + c(X)q_O(X) + q_C(X) + \text{PI}(X) \\ + (a(X) + \beta X + \gamma)(b(X) + \beta k_1 X + \gamma)(c(X) + \beta k_2 X + \gamma)z_1(X)\alpha \\ - (a(X) + \beta S_{\sigma 1}(X) + \gamma)(b(X) + \beta S_{\sigma 2}(X) + \gamma)(c(X) + \beta S_{\sigma 3}(X) + \gamma)z_1(\omega X)\alpha \\ + (z_1(X) - 1)L_1(X)\alpha^2 \\ + z_2(X)(1+\delta)(\varepsilon + q_K(X)c(X))(\varepsilon(1+\delta) + t(X) + \delta t(\omega X))\alpha^3 \\ - z_2(\omega X)(\varepsilon(1+\delta) + h_1(X) + \delta h_2(X))(\varepsilon(1+\delta) + h_2(X) + \delta h_1(\omega X))\alpha^3 \\ + (z_2(X) - 1)L_1(X)\alpha^4 \\ + q_T(X)t(X)\alpha^5 \end{pmatrix}$$

- Split $q(X)$ into 3 polynomials $q_{lo}(X)$, $q_{mid}(X)$, $q_{hi}(X)$ of degree at most $n+1$ such that:

$$q(X) = q_{lo}(X) + X^{n+2}q_{mid}(X) + X^{2n+4}q_{hi}(X)$$

- Compute $[q_{lo}(x)]_1$, $[q_{mid}(x)]_1$, $[q_{hi}(x)]_1$. Append them into `transcript`.

Round 5

- Compute the evaluation challenge $\xi \in \mathbb{F}$:

$$\xi = \text{Hash}(\text{transcript})$$

- Compute the opening evaluations and append them into `transcript`:

$$a(\xi), b(\xi), c(\xi), S_{\sigma_1}(\xi), S_{\sigma_2}(\xi), q_K(\xi), t(\xi), t(\omega\xi), \\ z_1(\omega\xi), z_2(\omega\xi), h_1(\omega\xi), h_2(\xi)$$

Round 6

- Compute the opening challenge $\eta \in \mathbb{F}$:

$$\eta = \text{Hash}(\text{transcript})$$

- Compute the linearization polynomial $r(X) \in \mathbb{F}_{<n+3}[x]$:

$$
\begin{aligned}
r(X) = & a(\xi)b(\xi)q_M(X) + a(\xi)q_L(X) + b(\xi)q_R(X) + c(\xi)q_O(X) + PI(\xi) + q_C(X) \\
& + \alpha[(a(\xi) + \beta\xi + \gamma)(b(\xi) + \beta k_1\xi + \gamma)(c(\xi) + \beta k_2\xi + \gamma)z_1(X) \\
& - (a(\xi) + \beta S_{\sigma 1}(\xi) + \gamma)(b(\xi) + \beta S_{\sigma 2}(\xi) + \gamma)(c(\xi) + \beta S_{\sigma 3}(X) + \gamma)z_1(\omega\xi)] \\
& + \alpha^2(z_1(X) - 1)L_1(\xi) \\
& + \alpha^3[z_2(X)(1+\delta)(\varepsilon + q_K(\xi)c(\xi))(\varepsilon(1+\delta) + t(\xi) + \delta t(\omega\xi)) \\
& - z_2(\omega\xi)(\varepsilon(1+\delta) + h_1(X) + \delta h_2(\xi))(\varepsilon(1+\delta) + h_2(\xi) + \delta h_1(\omega\xi))] \\
& + \alpha^4(z_2(X) - 1)L_1(\xi) \\
& + \alpha^5 q_T(X)t(\xi) \\
& - Z_H(\xi)(q_{lo}(X) + \xi^{n+2}q_{mid}(X) + \xi^{2n+4}q_{hi}(X))
\end{aligned}
$$

- Compute the opening proof polynomial $W_\xi(X) \in \mathbb{F}_{<n+2}[x]$

$$
W_\xi(X) = \frac{1}{X - \xi}\begin{pmatrix}
r(X) \\
+ \eta(a(X) - a(\xi)) \\
+ \eta^2(b(X) - b(\xi)) \\
+ \eta^3(c(X) - c(\xi)) \\
+ \eta^4(S_{\sigma 1}(X) - S_{\sigma 1}(\xi)) \\
+ \eta^5(S_{\sigma 2}(X) - S_{\sigma 2}(\xi)) \\
+ \eta^6(q_K(X) - q_K(\xi)) \\
+ \eta^7(h_2(X) - h_2(\xi)) \\
+ \eta^8(t(X) - t(\xi))
\end{pmatrix}
$$

- Compute the opening proof polynomial $W_{\omega\xi}(X) \in \mathbb{F}_{<n+2}[X]$:

$$
W_{\omega\xi}(X) = \frac{1}{X - \omega\xi}\begin{pmatrix}
t(X) - t(\omega\xi) \\
+ \eta(z_1(X) - z_1(\omega\xi)) \\
+ \eta^2(z_2(X) - z_2(\omega\xi)) \\
+ \eta^3(h_1(X) - h_1(\omega\xi))
\end{pmatrix}
$$

- Compute $[W_\xi(x)]_1$ and $[W_{\omega\xi}(x)]_1$.

Finally, we have the complete proof:

$$
\pi = \begin{pmatrix}
[a(x)]_1, [b(x)]_1, [c(x)]_1, \\
[t(x)]_1, [h_1(x)]_1, [h_2(x)]_1, [z_1(x)]_1, [z_2(x)]_1, \\
[q_{lo}(x)]_1, [q_{mid}(x)]_1, [q_{hi}(x)]_1, [W_\xi(x)]_1, [W_{\omega\xi}(x)]_1, \\
a(\xi), b(\xi), c(\xi), S_{\sigma 1}(\xi), S_{\sigma 2}(\xi), q_K(\xi), t(\xi), t(\omega\xi), \\
z_1(\omega\xi), z_2(\omega\xi), h_1(\omega\xi), h_2(\xi)
\end{pmatrix}
$$

5.3 Verify Process

- Validate all elements in π are valid.

- Validate $(w_i)_{i<l}$ are valid.
- Compute all the challenges $\beta, \gamma, \delta, \varepsilon, \alpha, \xi, \eta$.
- Compute the vanishing polynomial evaluation $Z_H(\xi) = \xi^n - 1$.
- Compute the first Lagrange polynomial evaluationn $L_1(\xi) = \frac{\xi^n - 1}{n(\xi - 1)}$
- Compute the public input polynomial evaluation $\mathrm{PI}(\xi) = \sum_{i=1}^{l} w_i L_i(\xi) = \sum_{i=1}^{l} \frac{w_i(\xi^n - 1)\omega^{i-1}}{n(\xi - \omega^{i-1})}$.
- Compute the first evaluation u:

$$\begin{aligned} u = \mathrm{PI}(\xi) &- \alpha(a(\xi) + \beta S_{\sigma 1}(\xi) + \gamma)(b(\xi) + \beta S_{\sigma 2}(\xi) + \gamma)(c(\xi) + \gamma) z_1(\omega\xi) - \alpha^2 L_1(\xi) \\ &- \alpha^3 z_2(\omega\xi)(\varepsilon(1+\delta) + \delta h_2(\xi))(\varepsilon(1+\delta) + h_2(\xi) + \delta h_1(\omega\xi)) - \alpha^4 L_1(\xi) \\ &- \eta a(\xi) - \eta^2 b(\xi) - \eta^3 c(\xi) - \eta^4 S_{\sigma 1}(\xi) - \eta^5 S_{\sigma 2}(\xi) - \eta^6 q_K(\xi) - \eta^7 h_2(\xi) - \eta^8 t(\xi) \end{aligned}$$

- Compute the first polynomial commitment $[U]_1$:

$$\begin{aligned} [U][U]_1 = &\, a(\xi)b(\xi)[q\_M(x)]_1 + a(\xi)[q\_L(x)]_1 + b(\xi)[q\_R(x)]_1 + c(\xi)[q\_O(x)]_1 + [q\_C(x)]_1 \\ &+ (\alpha(a(\xi) + \beta\xi + \gamma)(b(\xi) + \beta k_1 \xi + \gamma)(c(\xi) + \beta k_2 \xi + \gamma) + \alpha^2 L_1(\xi))[z\_1(x)]_1 \\ &- \alpha\beta z_1(\omega\xi)(a(\xi) + \beta S_{\sigma 1}(\xi) + \gamma)(b(\xi) + \beta S_{\sigma 2}(\xi) + \gamma)[S\_\sigma\ 3(x)]_1 \\ &+ (\alpha^3(1+\delta)(\varepsilon + q_K(\xi)c(\xi))(\varepsilon(1+\delta) + t(\xi) + \delta t(\omega\xi)) + \alpha^4 L_1(\xi))[z\_2(x)]_1 \\ &+ \alpha^5 t(\xi)[q_T(x)]_1 - Z_H(\xi)\left([q_{lo}(x)]_1 + \xi^{n+2}[q\_mid(x)]_1 + \xi^{2n+4}[q\_hi(x)]_1\right) \\ &+ \eta[a(x)]_1 + \eta^2[b(x)]_1 + \eta^3[c(x)]_1 + \eta^4[S\_\sigma 1(x)]_1 + \eta^5[S\_\sigma 2(x)]_1 \\ &+ \eta^6[q_K(x)]_1 + \eta^7[h\_2(x)]_1 + \eta^8[t(x)]_1 \end{aligned}$$

- Compute the second evaluation v:

$$v = -t(\omega\xi) - \eta z_1(\omega\xi) - \eta^2 z_2(\omega\xi) - \eta^3 h_1(\omega\xi)$$

- Compute the second polynomial commitment $[V]_1$:

$$[V]_1 = [t(x)]_1 + \eta[z_1(x)]_1 + \eta^2[z_2(x)]_1 + \eta^3[h_1(x)]_1$$

- Verify 2 KZG opening proof by pairing engine $e([\bullet]_1, [\bullet]_2)$:

$$\begin{aligned} e([W_\xi(x)]_1, [x]_2) &\stackrel{?}{=} e([U]_1 + u \cdot [1]_1 + \xi \cdot [W_\xi(x)]_1, [1]_2) \\ e([W_{\omega\xi}(x)]_1, [x]_2) &\stackrel{?}{=} e([V]_1 + v \cdot [1]_1 + \omega\xi \cdot [W_{\omega\xi}(x)]_1, [1]_2) \end{aligned}$$

- Verify m KZG opening proof at each identifier t_i by pairing engine $e([\bullet]_1, [\bullet]_2)$, where $1 \le i \le m$:

$$e([f_i(x)]_1, [x]_2) \stackrel{?}{=} e([t(x)]_1 - t_i \cdot [1]_1 + \omega^{i-1} \cdot [f_i(x)]_1, [1]_2)$$

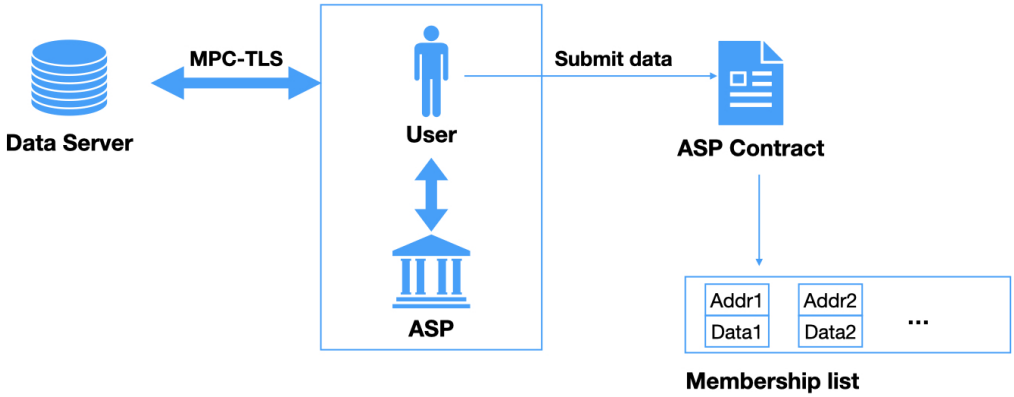
6. Association Set Provider

The Association Set Provider (ASP) mechanism allows a third party to oversee the membership list. Similar to an attestor, an ASP offers attestation services for end-users. The associated address, along with potential attested

and sealed data, is submitted to a designated ASP smart contract. In Vala, any entity can register as an ASP. The selection of which ASP to utilize depends on the choices made by end-users and Dapps.

The data category can be defined by the ASP, allowing support for a diverse range of potential data from web2, such as credit scores, KYC results, etc. For attesting any data obtained through the standard Transport Layer Security (TLS) protocol (e.g, HTTPS) and to accommodate a large volume of potential data, we recommend employing MPC-TLS style algorithms within the ASP. This approach, initially introduced by DECO and significantly improved by PADO, is detailed further in this paperhttps://eprint.iacr.org/2023/964.pdf [4]. Within this framework, users can prove to the attestor that the data indeed originates from the intended sources without leaking any other information.

We list the basic workflow in the following figure.



The inclusion of data in the membership list is discretionary. This flexibility arises from situations where the data entry might simply be binary (YES/NO). In such cases, the smart contract accepts addresses marked as YES, allowing the omission of unnecessary data entries. However, programmability can be introduced when the sealed data holds additional meanings. For instance, an ASP might attest a user's FICO score and store the encrypted score in the smart contract. Subsequently, Dapps can devise more adaptable withdrawal conditions. For example, users with higher FICO scores may be eligible to withdraw a larger quantity of tokens, whereas those with lower FICO scores might have access to only a smaller amount. This introduces a higher degree of flexibility for designing diverse applications.

7. Client-Side Acceleration Solution

In our quest to enhance user experience and efficiency in the Vala system, we've focused on client-side computational optimization. This optimization is crucial for reducing the computational burden on user devices and accelerating transaction and verification processes.

7.1 WebAssembly Integration

Our integration of WebAssembly (Wasm) allows for the execution of complex cryptographic proof generation processes directly in the user's browser. By compiling our Rust code into WebAssembly, we've significantly reduced dependency on centralized servers and improved system responsiveness and efficiency.

7.2 Local Computation Optimization

7.2.1 Transaction Decision Logic

In our UTXO model, each user's funds are represented as encrypted NOTES, each corresponding to a specific amount of assets (e.g., ETH). Our system intelligently selects the appropriate combination of NOTES to fulfill withdrawal requests. For instance, if a user has NOTES of 1 ETH, 1 ETH, 2 ETH, and 3 ETH and wishes to withdraw 2.5 ETH, our system automatically utilizes the 1 ETH, 1 ETH, and 2 ETH NOTES.

7.2.2 Generation of New Deposit Proofs

Following the utilization of certain NOTES for transactions, our system generates new deposit proofs corresponding to the remaining amount. Continuing the previous example, after spending the 1 ETH, 1 ETH, and 2 ETH NOTES, a new deposit proof for 1.5 ETH is created, ensuring secure and effective management of funds post-transaction.

7.2.3 Implementation in Chrome Plugin and iOS App

To achieve this computational optimization, we've implemented tailored strategies in our Chrome plugin and iOS app. The Chrome plugin optimizes the NOTE selection and new proof generation process, leveraging the browser's processing capabilities. In contrast, our iOS app employs multi-threading technology to accelerate these computations, taking full advantage of the high-performance capabilities of iOS devices.

7.3 Caching Strategies

We have implemented caching strategies to reduce redundant computations and network requests. Key data, such as parts of the Merkle Tree, are cached on the user device for quick retrieval in subsequent transactions or verifications, reducing network traffic and significantly enhancing overall system performance.

7.4 User Experience Enhancement

Lastly, we place a high emphasis on enhancing the user experience. This involves not only technical optimizations but also improvements in interface design. We ensure that the user interface is intuitive and the

transaction process is seamless. Real-time feedback and detailed error messages enhance user trust and satisfaction.

In summary, our client-side acceleration solution is a key strategy for enhancing the performance of the Vala system. Through these technological and methodological applications, we not only enhance the speed and efficiency of transactions but also optimize the user experience, making Vala a more powerful and user-friendly blockchain privacy platform.

8. what does it mean?

At ZKT Network, we're dedicated to realizing a grand vision: leading a technological revolution by balancing blockchain transaction privacy with global compliance standards. Our innovation extends beyond technical realms, exploring new frontiers in the balance of privacy and compliance.

Therefore, we warmly invite researchers and experts in the blockchain field to participate in our project, offering feedback and suggestions to advance this sector. ZKT Network looks forward to your professional insights and collaboration to develop a more comprehensive and effective blockchain privacy solution.

We also encourage community members to explore innovative applications based on our framework, as per their requirements. ZKT Network is eager to imagine the future with you, face challenges together, and build a more secure, compliant, and innovative blockchain world.

At ZKT Network, we believe that through collective wisdom and collaborative effort, we can achieve a perfect integration of privacy protection and compliance, creating a future filled with possibilities.

9. Summary

This paper thoroughly explores the approaches and practices for achieving a balance between privacy protection and regulatory compliance in blockchain technology. By incorporating innovative applications of honest address sets, zk-credit, and membership proof, we demonstrate how to maintain user privacy while adhering to regulatory standards. The focus extends beyond the technical and system architecture to include engineering practices and enhancements in user experience performance. Our aim is to establish a secure, efficient blockchain system that delivers an exceptional user experience. Representing a step forward, this second-generation privacy transaction protocol signifies a better tomorrow, where privacy is not just a feature – it's the norm.

1. Vitalik Buterin, Jacob Illum, Matthias Nadler, Fabian Schär, Ameen Soleimani (2023). "Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium" ↩
2. Luke Pearson, Joshua Fitzgerald, Héctor Masip, Marta Bellés-Muñoz & Jose Luis Muñoz-Tapia, (2022). "plookup: A simplified polynomial protocol for lookup tables" ↩
3. Ariel Gabizon, Zachary J. Williamson. (2020). "plonkup: PlonKup: Reconciling PlonK with plookup" ↩

4. Xiang Xie, Kang Yang, Xiao Wang, Yu Yu. (2023). "Lightweight Authentication of Web Data via Garble-Then-Prove." ↩