Goal

Trustlessly pay tokens to a large set of users, while keeping fees / network congestion low

Solution

Compress the payments into a single transaction, using merkle trees and off-chain data.

- Deposit the total token amount into a smart contract, along with a merkle tree root of the amounts allocated to each user
- Publish the full details of individual allowances off-chain
- Recipients can claim their allowance by submitting a merkle proof using off chain data

In essence, it acts like a one-to-many payment channel. Instead of opening a channel with every individual user, they can open a single channel (a “pool”), and pay many people at once.

Recurring Transactions

Although useful for one-off mass payments (e.g. an airdrop), the more interesting use-case is recurring payments to a group of users (dividends, royalties, etc). In order to support micro payments, you want a single user to be able to receive many payments over time, and then withdraw all of their funds in a single transaction.

To do this, the payer can update a previously created pool by depositing more tokens and submitting a new merkle tree root with the cumulative allowances of all users. When a user wants to withdraw, the contract will allow them to withdraw up to the amount allocated in their merkle proof, minus any they have already withdrawn. The user can use any valid merkle tree root (the older ones will just entitle them to a smaller allowance).

Features

- Payments are in ERC-20 tokens
- Each recipient can receive payments in unique amounts
- Cost is fixed, regardless of the number of recipients
- Payments cannot be taken back once settled
- Once a payment has settled, withdrawals are instant
- Recipients can withdraw multiple payments at once
- Recipients can withdraw any amount (does not need to be the full balance)
- When withdrawing, recipients can choose to send tokens to a different address. This way, the pool acts as a very basic on-chain wallet

Rules

- Each recipient’s allowance can only ever increase
- A recipient cannot withdraw more than their allowance
- Always enough tokens in the contract for everyone to withdraw their full allowance. New deposits could be verified using a merkle sum tree, which was proposed a way to ensure that Bitcoin exchanges weren’t running fractional reserves.

Attack Vectors

The ability to update a pool introduces some risk that needs to be mitigated. Normally, each user’s allowance should only ever increase. However, an attacker may try to submit a new merkle tree that increases their allowance, while reducing the allowances of others. To avoid this, pool updates could be subject to a challenge period (e.g. 1 day), to protect against the following scenarios:

Invalid Data

If an attacker tries to reduce an allowance, you can submit two merkle proofs (old and new) showing that an allowance went down.

Data Withholding

More likely, the attacker won't even publish details of the pool update off-chain, making it impossible to prove that allowances were reduced. In this case, the recipients need a way to mass exit or prevent the update from being accepted. Two possible solutions:

Exit Voting

When data is withheld, recipients who are paying attention will want to withdraw their tokens before the update is accepted. When doing so, they could set a flag, indicating that they are exiting due to a problem. If X% of recipients or allowances raise the flag, the pool is frozen, giving everyone time to withdraw. This way, not everyone has to be online.

Challenge

Alternatively, a challenger could post a bond, attesting that data was withheld. The pool owner would be required to submit the full set of merkle proofs, or else the pool would be frozen. If they successfully did so, they would receive the bond, and the update would be considered valid. One potential issue is that a challenger never knows if valid or invalid data is being withheld, so they may be unwilling to take the risk of posting a bond.

Outstanding Questions

- Any other attack vectors that are unaccounted for?
- Any better ways to prove that a balance has been reduced when data is being withheld?
- What should the protocol for off-chain data publishing be? JSON? REST API? Separate blockchain?
- Is there a simpler solution to achieve the same goal?

Alternative Solutions

This system is being explored as a solution for PROPS, where a large set of users are rewarded in a token on a regular basis, for their contributions in digital media applications.

While a similar result could potentially be achieved with a plasma chain or relay network, the aim is to design something simpler (for this specific use-case) that does not require operating a separate blockchain.

Any suggestions / feedback would be greatly appreciated. Thanks to Santiago for helping to refine.