Due to the inherent difficulty of executing smart contracts periodically, subscription and standing order payments have generally not been supported as protocols on the blockchain.

This difficulty arises from the fact that in protocols where nodes execute transactions, there is no guarantee that nodes can process all standing orders within a certain time frame if a large number of these transactions are reserved.

For instance, if three million people have reserved regular payments, completing the balance updates for three million ERC20 tokens within a single block is challenging.

However, if client-side validation or zero-knowledge proof (ZKP) execution is available, the burden on nodes will be reduced to merely posting data, such as a 256-bit hash value, without performing calculations. This only applies to the registration of subscriptions, as the calculation cost and DA (data availability) cost will be separate, but it still remains limited to that one-time initial cost.

To begin with, let's assume a Layer2 protocol where only the root of the Merkle Tree consisting of transaction hashes, like Intmax1 or Intmax2 by Albus, is recorded on the blockchain, and the proofs directly under this Merkle root are sent to the users or recipients. This article is applicable not only to Intmax but also to any protocol that utilizes client-side validation or ZKP. However, we will use Intmax as an example for easier explanation.

In these protocols, double-spending is prevented through non-inclusion proofs in the address list. On the client side, the outgoing transactions' balance reduction and incoming transactions' balance increase get proven using ZKP.

Simplifying Intmax1&2, the steps are as follows:

Algorithm 1

1. Senders send transactions.
2. The block producer aggregates transactions into a tree structure.
3. The block producer sends each proof to each sender.
4. The block producer creates a block with proofs signed by the address list and the Merkle root of the transactions.
5. Each sender sends the transaction and proof to each recipient.
6. The recipient uses ZKP to merge incoming transactions into their balance.
7. When the sender/recipient wants to exit assets to Layer1, they submit the balance and non-inclusion proof to the Layer1 smart contract.

Now, let's add the on-chain subscription registration tree to this setup.

The process of subscription registration and execution with Intmax2 is as follows:

Algorithm 2

1. Subscribers send new periodic execution transactions to the Layer1 (or Layer2) smart contract.
2. The smart contract updates the tree consisting of subscriptions and changes the root.
3. Simultaneously, the smart contract updates the tree consisting of subscriber addresses and changes its root.
4. The proof becomes trivial on-chain data for the subscriber.
5. The Layer2 block producer inserts the root of this tree near the root of the transaction tree.
6. The recipient aggregates the balance as in Algorithm 1. When performing an exit to Layer1, the address list is concatenated from both Algorithm 1 and Algorithm 2.

By doing this, it becomes possible to execute periodic payments for almost an unlimited number of accounts with minimal constant DA cost. The registration smart contract can also be implemented on Layer2, and the DA cost of Merkle proof can be minimized using either ZKP or blob.

Important Notes:

- Subscriptions will only be automatically activated if the sender's balance is sufficient. In Intmax2, invalid transactions from the past can be validated later through deposits, and subscriptions are also subject to this feature.
- The root is forced to enter the Layer2 blocks around the specified time in the Layer1 smart contract, so there should be no issues as long as the generation of Layer2 blocks does not stop for an extended period. To ensure this, it is recommended to distribute the Layer2 block producers.

- In Intmax1, the block producer had knowledge of the transaction’s validity, whereas in Intmax2, the validity of transactions is only known to the sender and recipient, making Intmax2 more suitable for this subscription protocol.
- Regardless of the version (Intmax1 or Intmax2), recipients still require proof that incoming transactions are valid, which means they either need to know the sender’s transaction history or receive proofs from the sender. Considering this aspect, it may be beneficial to create dedicated Subscription addresses where all payments are already known.