Money Over Mail Protocol (MOMP)

Jeevanjot Singh

jeevanjotsinghvital@gmail.com

www.momp.network

Abstract: MOMP. is an electronic assets transfer method from a **network wallet to** an **email address** without exposing the email address on the network itself. Relying on 3rd parties to manage email provides part of a solution but the main benefits of peer to peer cash transfer are lost when the wallet key pair system is also managed by a third party. I propose a solution with smart contract and hashing OTP(one time pin) with user credentials to verify email on top of the Blockchain network. There is no registration required for email (receiver) at first. Emails can withdraw their cash once they verify their email address with smart contract and map it with their wallet public address. Once they are verified, they will directly receive the cash on their public address, sent to their email. Verification is also controlled through smart contracts with verifying state variables .

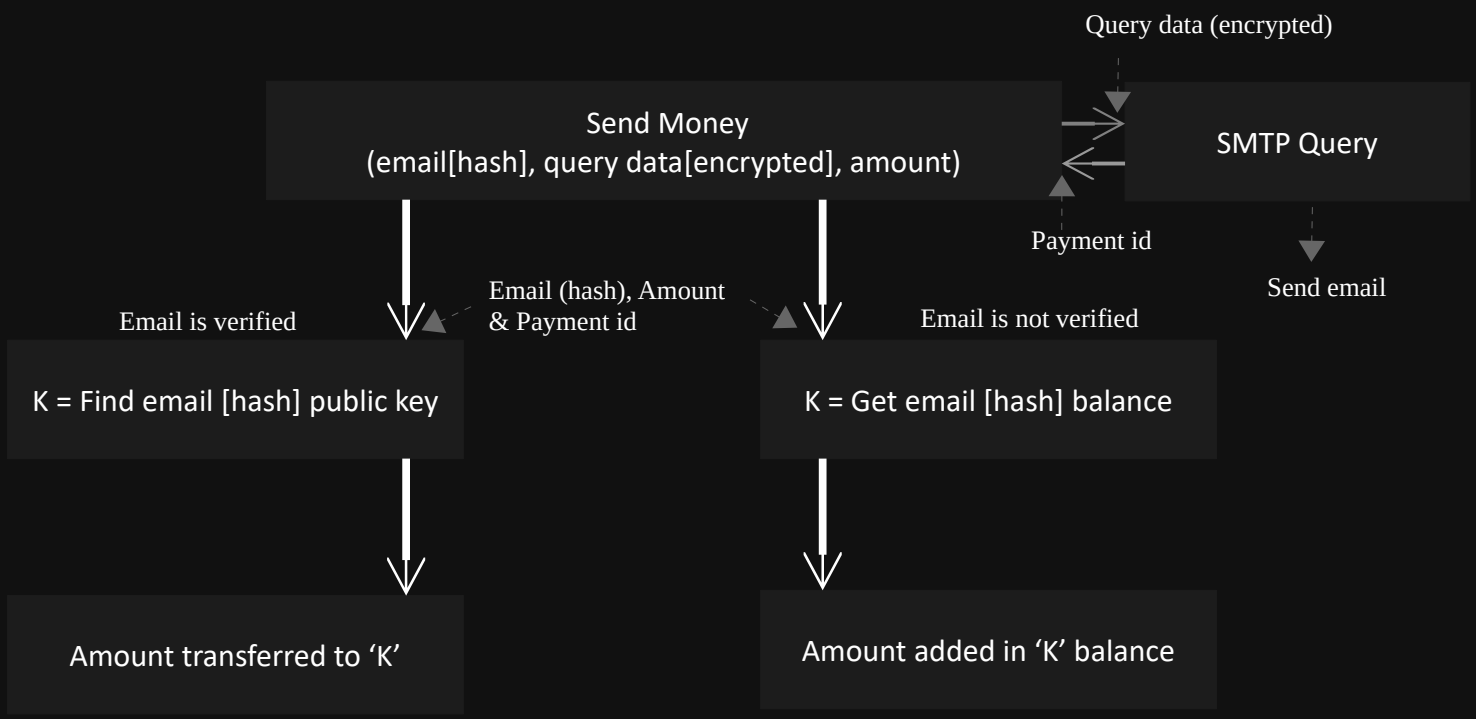
1. Introduction

Currently, there exists Cryptocurrency transfer methods that let users send digital assets through email but the cap is that they require trust on holding the user's private key as a receiver or require a private registration and they act like a middlemen.

MOMP provides a good balance between reducing the barriers of entrance for new users in the Cryptocurrency world and complete anonymous decentralized transactions. Using Money over Mail protocol, Any Cryptocurrency holder can send their virtual assets to any email (non wallet holder) without the need of a middle service. MOMP itself runs on the Blockchain smart contract from email registration and verification to asset transfer etc.

2. Transaction

Transactions for all the (registered or non registered) users are handled by the smart contract itself. Each transaction with money transfer is isolated with a unique ID generated from the "**Say Network**" module and is uniquely identifiable from the smart contract. If the receiver is not registered then the assets are added with existing balance under the beneficiary credentials in the smart contract or else they directly get spent to the beneficiary (receiver) .



Each transaction is defined with a unique payment id which is used to find the sender, **amount** (the value sent with the transaction) and the receiver. An unverified receiver holds its balance in the smart contract which can be withdrawn by the receiver only after verifying their **email (hash)** on top of the Blockchain (through smart contract) itself (as explained below) .

Send money (in its current state) also receives **query data (encrypted)** which is sent to the SMTP module from the "**Say Network**" and in return it gets a unique id .

3. “Say Network’s” SMTP Module

“Say Network” is an oracle service provider which provides verifiable data delivery to the network. SMTP module is provided by “Say Network” .

This solution provides email sending service on top of the Blockchain. Email data including receiver email, subject, email body etc. gets encrypted using a secret key through an API call. That encrypted data then passed to the Oracle contract of the “Say Network”, which is later processed by “Say engine” .

4. Claim Amount Back

e.g. When user **A** sends an amount to user **B**’s email and if user **B** has verified its email then the amount will be directly sent to user **B**’s wallet. However if user **B** has not registered its email and not verified with the OTP (One Time Pin) then the amount remains in the smart contract under user **B**’s credential that can be verified later (explained below) and user **B** (whether verified or not) will be notified on its email using SMTP module about their funds and to claim their amount .

However if user **B** does not verify its email under the smart contract in **15 days** (currently set time, from when the block time of the send money transaction) then the amount can be claimed by user **A** back to its wallet .

5. Withdraw Amount

Unverified users can verify their email address by providing a unique **one time pin** sent to their emails. Once Registered & verified, this allow users to withdraw their funds . MOMP network provides this feature on at the time of the verification or when user is verified but trying again.

Withdrawal process is only one time and once the user has verified its email with smart contract, The payment from that time uses a direct route of transfer to the verified public key

of the user. The smart contract do not hold any payment for that email once it is correctly verified with the public key.

6. Registration

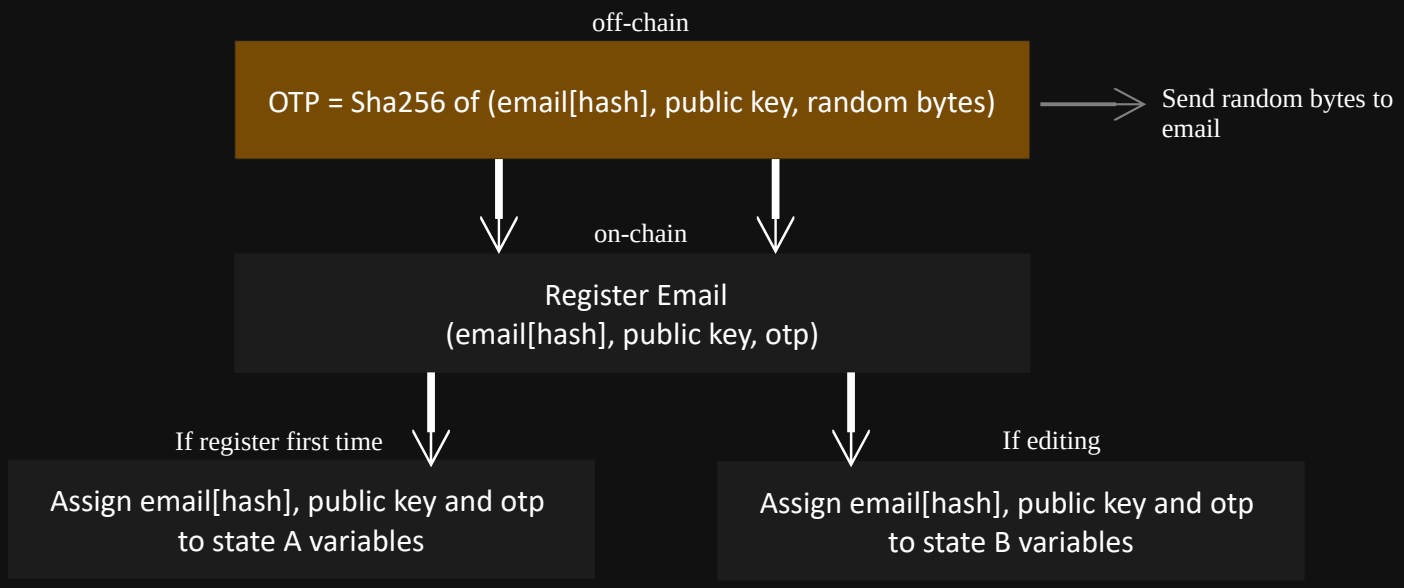
It is an only owner function & required to register & verify an email before the amount can be withdrawn. Registration process of **MOMP** is original and it is divided into 2 parts: **Register once**, **Edit registration** .

Register once

This function allow users to only register for one time. The purpose of this function is to create a separate stable state (once verified with main state) of user data for future transactions .

Edit registration

If the user is registered using first method then this function allow users to change their public keys for their emails in a separate non usable state and once the verification of email is done then the changes applies to the main state .



OTP is a combination of user credentials (public key, email) and random bytes .

Each of these functions have a separate verification method. Verification methods are explained below .

7. Verification

Verification is the most important step of MOMP. Emails are harder to verify on public networks because of their private communication nature .
e.g. If I want to perform a twitter name attestation on top of a Blockchain public network , then all I need is the client to verify their address through a public tweet which can be verified on top Blockchain using an oracle service (Which I have also worked on in the past & other social media attestations as well like Twitter, Discord, Github, etc.) but with emails, it is not the case. You cannot easily expose data attached to the email publicly to verify it on chain (more challenges) .

So, I propose a **Two-way verification method**, In which outside hashed data (out from the network) to match with the data that will be used inside smart contract (in the network).

Privacy with transactions: Also I added a thought of avoiding email addresses getting exposed on public networks hence, hashed emails. Hashed emails are digested email addresses with cryptography hash function **Sha256**. Email hashes are always used for performing transactions inside **MOMP network** .

Verification is done by **comparing hash:** created at the time of **registration** (by owner), with the one created on top of Blockchain at the time of **verification** (by user).

Registration

Sha256 of sha256(Email)+public key+OTP

Verification

Sha256 of sha256(Email)+Find Public key (Email)+OTP

Working: Registered email is verified by providing email (hash) & the original OTP(random bytes) which is to be received separately on the email. A User is required to run function for verification by providing the received OTP. Then inside the smart contract, the public key registered with email perform same function that was performed off-chain when registering and verifies if the random bytes (received on email) are correct and the email is indeed the user's. The same state variables that are used to verify the hashed string will be used to perform future transactions. This verifies that the same values are getting used. Hiding email in public network and requirement of OTP Original provide strength for the security of this protocol. So if you are not the owner of the email, then the email cannot be verified. Registration function require owner privileges but verification does not and run directly by the user. This creates two-way verification possible .

8. Verification Again

A user might want to change its public key later with the verified email address. It should be done again with such methods so that the results can be aligned with the request done by the owner of the requested email.

Working: Verification again functions same as verification, Where a new registration (On a separate / temporary state) is required and a separate verification method should confirm the email and write on the original state (**transaction state**).

Making it completely trust-less: After the first time registration, The most important and a **new requirement** for this function is that the function should be run by the owner's current verified (currently saved public key on original state) account, This now provide a trust-less view where the owner cannot (In any way) add its own address under your email.

[This document can receive updates in future]