Computer Security Symposium 2020
26 - 29 October 2020

A Study on Registered Email System Using Blockchain

Kouame Michel Robin[1,a)] Inaba Hiroyuki[1,b)]

Abstract: With the recent expansion of the Internet, the electronic mail system has been widely promoted. With various proprietary and opensource systems developed, registration feature is not offered. In this paper, we present a new way of providing registration in email systems. We have implemented a blockchain-based email protocol built upon the Ethereum public blockchain and the current email system. The protocol tracks and traces transmitted message between parties. Furthermore, the protocol encounters repudiation by tracking the receivers' actions on the received email, while ensuring confidentiality, integrity and authentication. A detailed analysis of the security is presented.

Keywords: Blockchain, Email Security, Smart Contract, Ethereum

1. Introduction

Email systems are one of the most used services over the Internet. Even if we have many solutions, they lack the registration service. We suppose that an important email is sent between two parties A and B, and B deletes this message or avoid opening it for a long time. Then, when A complains, B answers that he/she did not receive such a message. The current study presents a solution to resolve this conflict without any third party.

Our solution is based on the blockchain distributed computing system like in [2]. We implement a new protocol built upon the Ethereum public blockchain to offer this registration service [1]. The protocol allows the email originator to track the message transmission through the Internet. Moreover, it tracks the email recipient actions on the received message and detects a delay in the reading. The rest of the paper is organized as follows: we introduce our new protocol in section II. The implementation details are described in section III. The security analysis of our design are performed in section IV. Finally, we present the conclusions in Section V.

2. Background

The Message Disposition Notification protocol is a solution for confirming the existence of a designated email message at its destination after transmission. The protocol is specified in the RFC 1459. It describes the MIME content type to be used by a mail user agent(MUA) or any electronic mail gateway for reporting the existence of an email message after this one has been successfully delivered. This protocol provides a tracking functionality for email messages between parties. Even if the idea behind the design shows some interests, this solution seems to have some limitations. The first limitation is about users privacy. In fact, recalling our example where an email is sent from A and B, A will request the notification of receipt from B. But, A also does not want B to know that he is using this feature. The second limitation concerns the timeout feature. MDN is used to notify the sender about the actions on the receiver side, that is if the message is displayed, printed, deleted or if the recipient refuse to proceed the MDN request. But in this condition, the MDN does not provide a result for the case in which the message is ignored by the receiver for a certain time(a timeout parameter). As a third point, MDN messages could be falcified by bad users to declare false diposition. Finally, some agents could send massive unsolicitated MDN messages that could lead to a denial of service attack.

Having exposed the above arguments, this leads in this current study to design a delivery notification protocol using the blockchain. Here are our main contributions:

- the design of a Blockchain-based email delivery notification protocol
- Ensuring user privacy better than the MDN protocol
- We provide a timeout feature for managing a delay in message reading
- We garantee using the blockchain payment service that the notification request is charged for avoiding malicious behavior by the users

3. Proposed Protocol

3.1 Overview

3.1.1 Protocol Architecture

Fig. 1 shows the operating diagram. The system includes

1 Kyoto Institute of Technology, 京都工芸繊維大学
a) kouame15@sec.is.kit.ac.jp
b) inaba@kit.ac.jp

a user interface which is handled by the sender and the receivers. During its transmission, the email reaches first the sender's email (SMTP) server. Then the sender's SMTP forwards it to the Internet relay servers until the last one conveys it to the receiver's email server. Each server handles the email processes the registration.

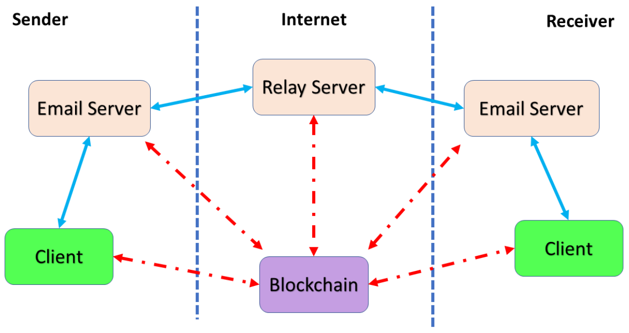

Fig. 1: Protocol overview.

3.1.2 Architecture of Smart Contracts

The **EmailRegistrationMaster** contract orders the **EmailRegistrationFactory** contract to create and deploy the **EmailRegistration** contract for a designated email. In this design, an email sent to many recipients is handled with only one instance of **EmailRegistration** contract. A status list is kept for each element of the list of recipients. The **EmailRegistration** contract owner field is set to the sender's blockchain address.

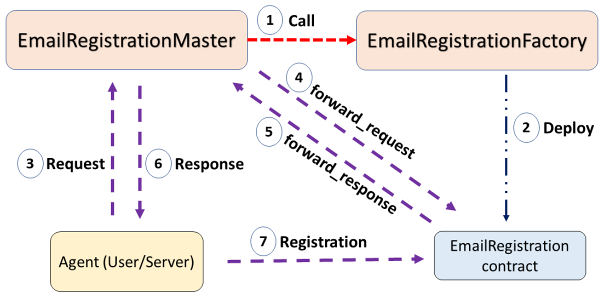

Fig. 2: Protocol Architecture.

3.2 Assumptions

The sender's SMTP server, the receivers SMTP servers, and at least one of the intermediate relay servers have the capability to speak the current protocol. They have each a blockchain account. Relay servers unable to speak the protocol will ignore the related feature. The sender's email server's blockchain address is known by sender's client. The following provides a detailed description of the diagram on Fig. 2.

3.3 Initialization and Email Transmission

The sender handles the client to set the parameters values for the email system and the blockchain contracts.
Step 1: the client creates and deployes the registration contract through the Master contract which replies with the address of the registration contract.
Step 2: the client builds the header of the email by including the registration contract address and the local ID generated with the timestamp and a sequence number.
Step 3: the client sends the email to the sender's email server. The email header is encrypted with the sender's server blockchain address public key. The sender pays the fee for the registration contract deployment.

3.4 Registration

It is mainly managed in the **EmailRegistration** contract.
Step 1: the sender's email server extracts the contract address from the email header and uses it to build the registration transaction. This transaction's price is calculated in the protocol, and paid by the server of the sender.
Step 2: using the transaction ID, the contract sends the payment status to the sender's client who reimburses the server. This cost is finally charged by the server to the sender.
Step 3: the server updates the header with the new **EmailRegistration** contract address. After the SMTP QUIT command between the sender's server and the relay server, the sender's server updates the server list in the **EmailRegistration** contract with this relay server blockchain and email addresses. The relay server generates the transaction and pays the fee. Like in **Step 2**, the sender receives a notification of payment to reimburse.

3.5 Receivers Action Tracking

The receiver client downloads the email from its email server. This email server detects the receivers actions on the email and sends the notification information to the **EmailRegistration** contract.

3.6 Email Status Tracking

The sender client receives notifications from the blockchain about the registration contract and its associated email. When the receiver does not download the email for a long time, meaning after an amount of time set as timeout parameter, the receiver's email server sends the status to the registration contract.

4. Implementation

4.1 User Interface

The sender handles the web client to set the values of his/her blockchain and email addresses along with the receivers email addresses, the timeout, and the email data. Like the SMTP protocol, the timeout value can be set less than ten days. The client sends the email to the sender's SMTP server via the SMTP protocol. The receiver also uses the client to access its mailbox through IMAP or POP protocol. In addition to calling the smart contracts methods, the client generate a unique ID for each email.

4.2 Email Server

Procmail is an email processing utility for Unix systems.

At the MDA (Mail Delivery Agent), Procmail operates before emails reach the user client. It filters the received emails and extract in the header the values of the parameters used to build and send the blockchain transaction **EmailRegistration** contract address.

4.3 Ethereum Public Blockchain

The Ethereum public blockchain offers an efficient smart contracts eexecuting for interacting with the client or email server. Ethereum transactions generated by the sender's client or the relay server execute contracts methods and return values. The protocol calculates the transaction price as the product of the network gas price and the method's gas cost. This transaction is built using the server blockchain address, along with the contract address, the method, the values of the method's parameters and the gas price.

4.4 The Blockchain Smart Contracts

The **EmailRegistrationMaster** contract keeps a mapping of **EmailRegistration** contract address with this contract owner's blockchain address. These information are encrypted with the owner's blockchain private key and provided with that owner's signature on them. For the registration, the server sends a request to the **EmailRegistrationMaster** contract, which forwards it to the **EmailRegistration** contract. The **EmailRegistration** contract verifies the existence of the server's blockchain address in the list of agents allowed to perform the registration and replies to the **EmailRegistrationMaster** contract. This last contract responds with the **EmailRegistration** address and the hash of this address both encrypted with the server's public key. Then, the server computes this address, accesses the **EmailRegistration** contract and performs the registration.

5. Security Analysis

5.1 Illegitimate registration

The attacker's wants to intercept the information sent between two legitimate servers or client and his/her email server in order to perform false registration and prevent the legitimate agent (server or user) to register. But, any call to the **EmailRegistration** contract requires the caller authenticate through the **EmailRegistrationMaster** contract. This prevents the illegitimate user from acting.

5.2 Users privacy in the public blockchain

To access the **EmailRegistration** contract, the relay server or client must send a request to the **EmailRegistrationMaster** contract. This public contract requires the user to authenticate as the owner of the **EmailRegistration** contract or as a member of the list of servers or clients allowed to perform a registration. This preserves the privacy of the sender and the receivers information.

6. Conclusion

The protocol adds registration feature to the current email system by using the public blockchain. It the sender with email tracking and status notification of the receiver's actions. Moreover, it offers confidentiality, integrity and user authentication. A future study could consider a third party through dispute resolving. The email sender pays around 1 USD.

References

[1] P. CUI , J. DIXON, U. GUIN, D. Dimase, "A Blockchain-Based Framework for Supply Chain Provenance," 2019 IEEE vol. 7, pp. 113–125, October 2019.

[2] T. Sanda and H. Inaba, "Proposal of new Authentcation Method in wi-fi Access using bitcoin 2.0," 2016 IEEE 5th Global Conference on Consumer Electronics (GCCE), pp. 459–463, Japan, Sept. 2016.

[3] Ethereum:(online), available from ⟨http://ethereum.org⟩ (2020.08.17).

[4] Ether price:(online), available from ⟨https://ethereumprice.org⟩ (2020.08.17).