A Cryptocurrency-based E-mail System for SPAM Control

Shafiya Afzal Sheikh[1], M. Tariq Banday[2]
Department of Electronics and Instrumentation Technology
University of Kashmir, Srinagar, India

***Abstract*—Sending bulk e-mail is commercially cheap and technically easy, making it profitable for spammers, even if a tiny percentage of recipients falls for the attacks or turn into customers. Some researchers have proposed making e-mail paid so that sending bulk e-mail becomes expensive, making spamming unprofitable and a futile exercise unless many victims respond to spam. On the other hand, the small sending fee is negligible for legitimate e-mail users. Making e-mail paid is a challenging task if implemented using a conventional payment system or developing new cryptocurrencies. Traditional payment systems are challenging to integrate with e-mail systems, and new cryptocurrencies will have challenges in adoption by users on the required scale. This work proposes using cryptocurrency payments to make e-mail senders pay for sending an e-mail without creating a new cryptocurrency or a new blockchain. In the proposed system, the recipients of the e-mail can collect the payments and use the collected revenues to send e-mail messages or even sell them on an exchange. The proposed solution has been implemented using Ropsten, an Ethereum Test Network and tested using enhanced E-mail Client and Server software.**

Keywords*—*E-mail; SPAM; blockchain; cryptocurrency; Ethereum

I. Introduction

E-mail spamming is one of the critical technical challenges the cyber community faces, causing problems on various fronts. In 2019, 28.5% of total global e-mail traffic was SPAM and was above 45% of total e-mail traffic in 2018 [1]. This huge percentage of SPAM traffic forces e-mail service providers and users to invest hugely in anti-spam technologies and infrastructure, which is a substantial commercial loss globally. These SPAM e-mails take up a considerable amount of storage space on both the e-mail servers and clients. In addition to this, SPAM e-mails pose a multitude of threats to e-mail users by tricking them into visiting malicious websites where they get usually cheated and suffer financial loses. The spammers use phishing, credential phishing, spear phishing, whaling, blackmailing, etc., to harm the recipients in various ways. The SPAM e-mail can also spread and install viruses, rootkits, exploits or other malicious software resulting in data leakage, including passwords, credit card details, etc. and may even compromise the underlying computer software.

Cryptography [2] is being extensively used in securing online communications, including e-mail messaging. Many communication protocols and security features used in the e-mail messaging system rely on cryptographic techniques. These techniques help make e-mail communication safe against the attacks like unauthorized access, spoofing [3], spamming [4], phishing [5], etc. Cryptographic techniques are also used in various e-mail encryption techniques to send encrypted e-mail to prevent unauthorized access to e-mail message while in transit over a network. The most common protocols and standards which are used for this purpose include GNU Privacy Guard (GPG) [6], Pretty Good Privacy (PGP) [7], Secure Multipurpose Internet Mail Extensions (S/MIME) [8], Privacy-Enhanced Mail (PEM) [9], OpenPGP [10], etc. The Transport Layer Security (TLS) also falls in this category of techniques, which helps encrypt e-mail communication at the transport layer.

Domain Key Identified Mail (DKIM) [11] is a popular cryptography-based e-mail authentication technique which attaches Digital Signatures to outgoing e-mail messages from a domain name. The Digital Signatures are generated by the sending server using Asymmetric Key Cryptographic technique. The recipient can verify them to ensure the incoming e-mail has arrived from the domain name it claims. This helps counter e-mail spoofing, which can result in e-mail SPAM and various email-based phishing attacks.

In combination with blockchain [12] technology, cryptography has revolutionized the payment systems through cryptocurrency. A cryptocurrency is a form of a digital asset that acts as a medium of exchange. The ownership records of assets are stored in a blockchain that is a distributed and decentralized ledger. Cryptocurrencies use modern and secure encryption techniques to secure transactions, verify transactions, and generate new assets. There is no central authority or a trusted party in between the individuals involved in a transaction. The cryptocurrency system works entirely on top of robust cryptographic techniques. Some of the most popular and widely used Cryptocurrencies and blockchain applications are Bitcoin [13], Ethereum [14], Litecoin [15], Chainlink [16], Ripple [17], Stellar Lumens [18] and many more.

Researchers have proposed making e-mail senders pay for sending an e-mail, using tiny amounts, to make sending e-mail for spammers an expensive endeavour. However, there is no robust payment solution to make it possible. Cryptocurrencies can play a vital role in making such proposed system work in reality and can help control e-mail SPAM by making it an expensive and a futile effort for spammers.

A. Contribution

In an attempt to control SPAM by making it expensive, this work proposes the use of existing cryptocurrency and blockchain of payments for sending e-mail messages, without

the need to set up an entirely new payment network. The proposed system has been implemented and demonstrated on Ropsten Ethereum Test Network. It includes implementing a new communication protocol to distribute Wallet Addresses and support creating and verifying cryptocurrency transactions within the existing e-mail infrastructure. The solution's implementation and demonstration include developing a primary E-mail Server with support for the proposed protocol, making and verifying the transaction. The implementation also includes enhancing an open-source e-mail client to create the transactions and attaching them to e-mail messages.

II. Literature Review

Adam Back's Hashcash [19] proposed a proof-of-work-based solution to counter automated abuse of Internet services like e-mail, HTML form submissions, etc. The said solution requires the user to perform some function on the system with moderate CPU requirement before using a service like sending an e-mail. In an e-mail system, this technique ensures that the sender machine has performed some computation and utilized a moderate CPU resource before sending out the e-mail. This computation hardly makes any difference for a legitimate e-mail sender but is extremely expensive for spammers who send many SPAM e-mails within a small-time duration. This technique dramatically reduces the number of e-mails spammers can send within a given amount of time using limited resources. The major drawback to this technique is that spammers will prefer to attack computer users and online servers, control them and use them to process and send e-mails, rather than use their hardware to do the processing. That way, the spammers do not have to invest anything other than hacking into other people's resources and using them for sending their SPAM.

David A. Turner and Ni Deng [20] proposed a solution of payment-based e-mail using Lightweight Currency Protocol. Their proposed solution suggests enhancing the SMTP protocol and includes the payment and payment verification processes into the SMTP protocol. They propose that e-mail service providers will issue their currencies for payments, which will lead to the creation of an unlimited number of currencies that become impossible to manage and lead to problems while performing inter-server mail exchanges. Sending e-mails to different e-mail servers will require purchasing or acquiring a large number of different currencies. Another drawback in their proposed solution is that payments take time to process and verify and including the two processes into the SMTP transaction will make the transaction take too long. This will result in highly reducing the performance of the SMTP servers. Furthermore, changing an existing, widely used protocol is not feasible in a real-world situation. Therefore, the payment transaction and transaction verifications should be independent of the widely used SMTP protocol.

K. Nakayama, Y. Moriyama and C. Oshima [21] have proposed an algorithm named $SAGA_{BC,}$ which uses blockchain technology to control the e-mail SPAM. They propose to make the Sender pay a processing fee to send an e-mail using a new Cryptocurrency which they name as "Mail Send Coin". The fee paid by the senders is later refunded if the e-mail messages are received correctly by the e-mail recipients. However, their claim to develop and implement an entirely new cryptocurrency for their proposed solution is technically impractical. The cryptocurrency will require mining which should generate returns for the miners. There is no reason why for miners to be interested in investing in mining their proposed cryptocurrency. Another drawback in their proposed system is that refunds are never guaranteed in any cryptocurrency, and it entirely depends upon the recipients to make any refunds. There is no central authority or trusted intermediary in cryptocurrency networks who can guarantee or force a refund in successful e-mail delivery. Furthermore, the authors have not proposed any practical methodology for implementing payment transactions or transaction verification by the recipients.

Likewise, the Credo project's concept is to use Credo Tokens as a payment method for an e-mail service provided by BitBounce. The service makes e-mail senders, who are not in the recipient's contact list, pay for sending e-mails. The non-whitelisted e-mail senders get a payment request when they try to send an e-mail to an e-mail account on the BitBounce Server. However, the credo cryptocurrency is linked directly to the BitBounce e-mail service. There is no information about if any other e-mail service providers can adopt this technology or the cryptocurrency [22].

These researchers and service providers have tried to provide solutions which could help make e-mail messaging paid and control e-mail SPAM. Due to the limitations in their proposed solutions or lack of practical methodology, this work offers a solution that overcomes these shortcomings and proposes a design implemented and tested, using existing cryptocurrency systems without any widely accepted changes e-mail communication protocols.

III. Proposed Solution

The work presented in this paper proposes a methodology that uses the existing cryptocurrency as a mode of payment for sending e-mail messages. The Sender of the e-mail, either the E-mail Client or the E-mail Server, makes a payment in advance, for the e-mail message it sends to the recipient. The recipient will get the e-mail message in the inbox, only if the payment is verified successfully. The payment transaction can be made by the sending Mail Transfer Agent (MTA) on behalf the domain users or by the sending Mail User Agent (MUA), depending upon the payment policies and network configuration of the e-mail infrastructure. The proposed solution is comprised of multiple processes viz: (a) Wallet Creation, (b) Wallet Address Distribution, (c) Payment and Stamp Creation, (d) Sending the e-mail, (e) Transaction Verification and (f) Delivering E-mail to MUA.

In the proposed solution, a Cryptocurrency is used for payment processing in which the e-mail sender is required to make a small cryptocurrency payment for sending an e-mail. The small fee is negligible for legitimate users who send a limited number of e-mails. However, the cost will accumulate to form a massive number for spammers who operate by sending out a large number of e-mails in a short time, making spamming expensive and useless. The payment is made using an existing cryptocurrency transaction, and the payment has

been made a prerequisite for delivering the e-mail to the inbox of recipients. The work also proposes a new wallet distribution protocol. With the help of which the Sending MTA or the sending MUA (Email Sender) after opening a connection with the recipient MTA asks for the list of supported cryptocurrencies. The recipient MTA responds with the list of supported cryptocurrencies. The E-mail Sender then requests for the wallet address of the recipient e-mail address on that domain to which the recipient MTA responds with the wallet address of the e-mail address. The E-mail Sender makes a cryptocurrency payment favouring the wallet and attaches the transaction hash of the payment to the e-mail header and sends the e-mail to the recipient MTA. This research names the transaction hash in the e-mail header as "EmailSTAMP". The 'EmailSTAMP' in the e-mail header can be thought of as a postal stamp on a letter's envelope, which implies that the Sender has obtained the stamp by making a payment.

On the receiving side, the 'EmailSTAMP' from the e-mail header is extracted, and the transaction is verified on the blockchain. On successful transaction verification, the e-mail is delivered to the inbox of the recipient e-mail address. If the transaction verification fails or a transaction hash is not found in the e-mail header, the e-mail is categorized as SPAM by the recipient MTA. This proposed solution makes sure that the Sender makes a tiny payment for each e-mail it sends, favouring the recipient or the recipient MTA for successfully delivering e-mails to the recipient's inbox. Fig. 1 show a sequence diagram of the proposed solution.

The various steps and processes involved in the proposed solutions are discussed below in detail.

A. Wallet Creation

The e-mail server creates wallet addresses for all the e-mail accounts present on the e-mail server and stores the private keys securely. The E-mail Server may even create wallets for multiple cryptocurrencies for every e-mail account. The wallets are used to receive payments from the e-mail senders. The wallets can be created immediately when creating the e-mail accounts or whenever an e-mail server requests one. The E-mail Client can request the Private Key from the E-mail Server, after the proper authentication. In an alternate implementation, instead of the E-mail Server, the E-mail Client generates and stores the secret keys and wallet addresses, and shares the E-mail Server's wallet address. In such cases, the payment transaction is made by the E-mail Client itself and attached to the e-mail header before submitting it to the MTA for sending. Fig. 2 shows the flowchart of creating a wallet by the E-mail Server when an E-mail Sender requests a wallet.

As shown in Fig. 2, the Mail Server generates a wallet and returns it to an E-mail Sender for a non-existent e-mail address to counter the problem of e-mail address harvesting. E-mail spammers use various techniques to collect valid and working e-mail addresses to spam them and ensure that most of the e-mail being sent is received by e-mail addresses that exist and are active. One of the methods of collecting e-mail addresses is guessing and cleaning. In this method, the spammers guess e-mail addresses and send them e-mail messages. If the e-mail address is invalid and receives a bounce mail, they remove the e-mail address from the list and keep the ones that did not respond with a bounce mail. This way, they can collect working e-mail addresses and filter out invalid ones. In the proposed technique, if the sending server queries for the wallet address of invalid e-mail addresses, the e-mail server still generates a wallet at runtime and return that to the sending server. This way, instead of returning an invalid-email-address message, the e-mail server will respond positively with a wallet address. If e-mail messages are afterwards received on such, non-existent e-mail addresses, the Server can still collect the payments made and discard the e-mail message silently. This way, the sending e-mail server is made to pay for e-mail address harvesting using guessing and cleaning technique.

B. Identify the Headings

A communication protocol has been proposed, which E-mail Senders and E-mail Recipients use to request and share wallet addresses of e-mail addresses. The sending MTA or the sending MUA (Email Sender) asks the recipient MTA for an e-mail address's wallet address. The recipient MTA responds with a wallet address. The E-mail Sender verifies the wallet address for correct format. If the wallet address is confirmed to be a valid address, the E-mail Sender will use that wallet address to make payment for the e-mail message. Fig. 3 shows an example of the protocol for distribution of wallets against e-mail addresses after establishing a connection. As shown in Fig. 3, R denotes the Recipient Server, and S represents the Sending MTA or MUA. After establishing a connection with the Recipient Server, the E-mail Sender receives a welcome message with a 220-status message. The E-mail Sender issues a command LIST CURRENCY. The Recipient Server responds with a 220-status message and a list of supported cryptocurrencies. The E-mail Sender issues a command WALLET followed by the cryptocurrency preference, a colon and an e-mail address to send the e-mail. According to the Sender's selected cryptocurrency, the Recipient Server responds with a 220-status followed by a wallet address. After receiving the wallet address, the E-mail Sender closes the connection.

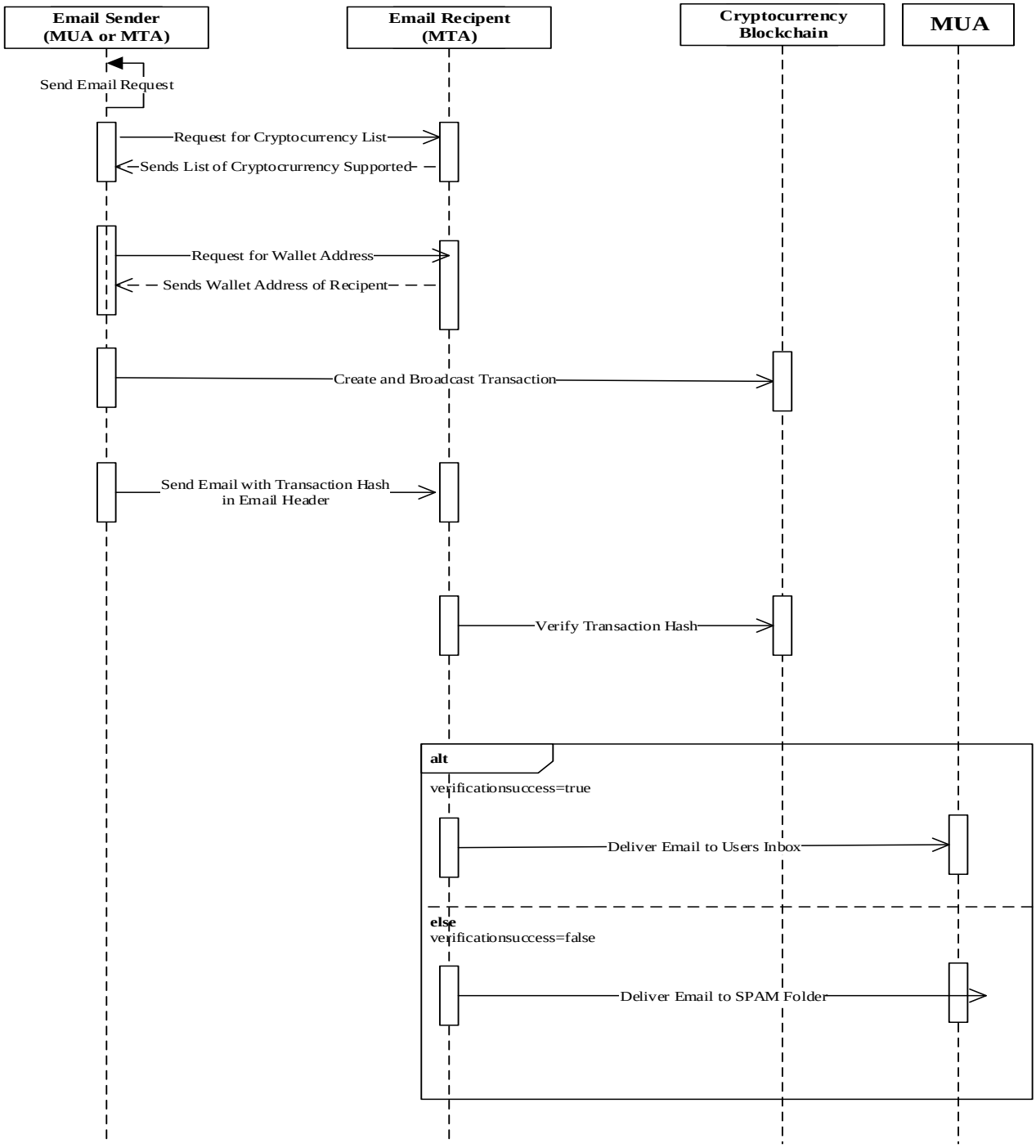

Fig. 1. Sequence Diagram of the Proposed System.

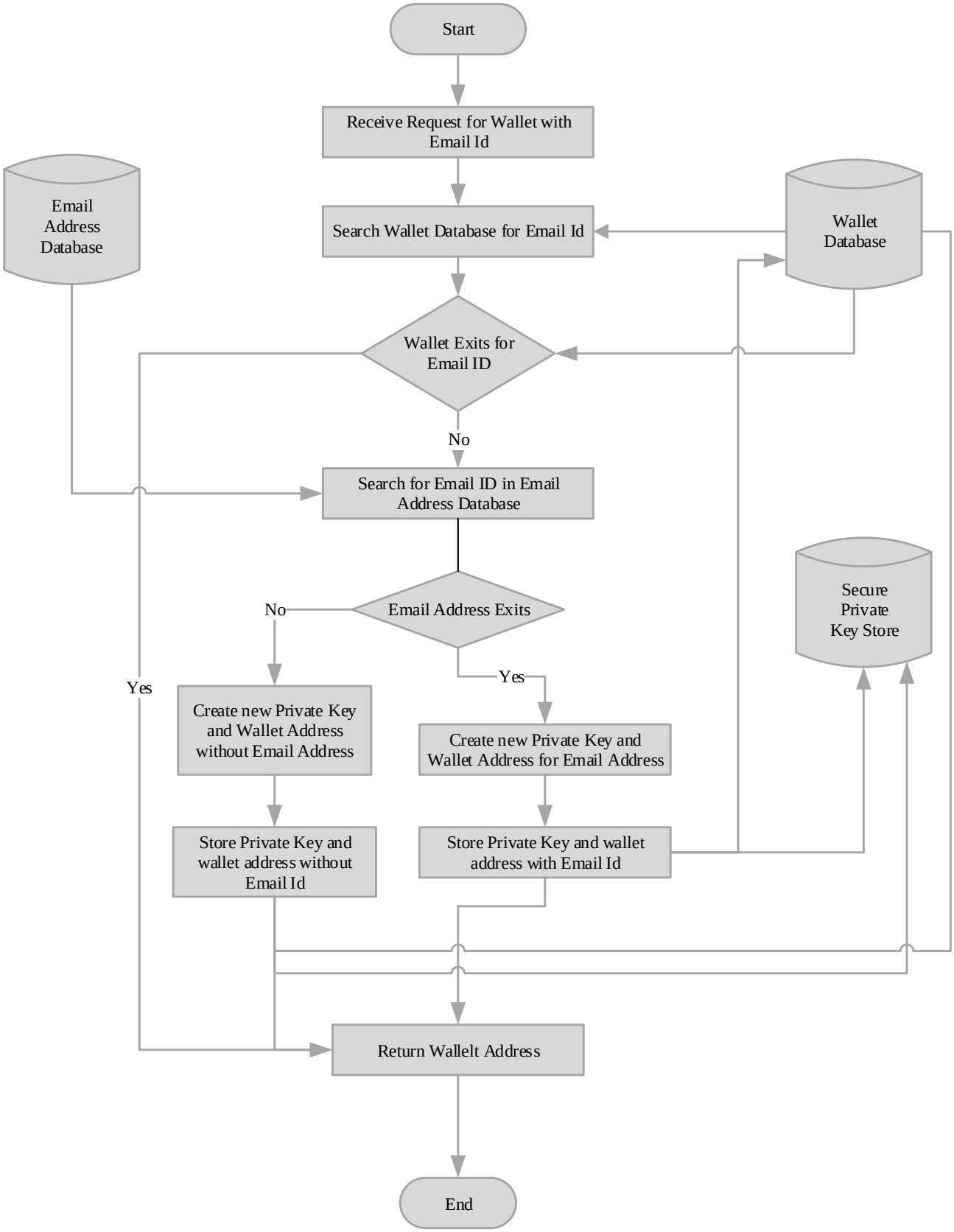

Fig. 2. Flowchart Showing the Creation and Returning of a Cryptocurrency Wallet by the E-mail Sender.

R: 220 smtp.recipient-domain.com ESMTP Postfix
S: LIST CURRENCY
R: BITCOIN ETHER DOGE.
S: WALLET DOGE:some-user@recipient-domain.com
R: 220 DB1Xu2kgdkgu83UWxFE3r9hJiG65FaC003D

Fig. 3. An Example of a New Wallet Distribution Protocol.

C. Cryptocurrency Payment and 'EmailSTAMP'Creation

In the proposed solution, the Sending MTA or the Sending MUA (Email Sender) is enhanced to generate a cryptocurrency transaction. The transaction is made up of an input wallet address, the wallet address of the e-mail recipient and the amount to be paid. The E-mail Sender concatenates the FROM e-mail address, the recipient e-mail address, the amount and the e-mail subject, together and calculates an md5 hash of the resulting string, named "verification-hash". The verification-hash is then attached to the cryptocurrency transaction and is afterwards used to verify if the payment transaction in the *'EmailSTAMP'* applies to the e-mail attached to it re-calculating the verification-hash on the recipient side. The entire transaction is signed by the private wallet key of the E-mail Sender. The transaction hash of the transaction is calculated, and the transaction is broadcasted into the cryptocurrency blockchain network. Fig. 4 shows the structure of a cryptocurrency transaction that can be used as a proof of payment for a given e-mail message. The same transaction cannot be used for any other e-mail messages because of the "verification-hash".

D. E-mail Message Submission

The e-mail server creates the e-mail address in the usual way and adds an extra header, EmailSTAMP, to the e-mail headers. The 'EmailSTAMP' header field contains the cryptocurrency name and the transaction hash, as a proof of payment. The e-mail is then sent to the Recipient Server for delivery.

A sample set of headers is shows in Fig. 5, wherein the 'EmailSTAMP' header field includes the name of the cryptocurrency and the transaction hash. After adding the 'EmailSTAMP' header, the Sending Server can send the e-mail to the Recipient Server in a regular manner. The Recipient Server can use the header information to check for the transaction information on the relevant blockchain.

E. Transaction Verification

The recipient server on receiving the e-mail checks the e-mail header extracts the stamp which contains the transaction hash. It queries the blockchain for the correctness of the transaction hash and gets the information about the transaction. If the transaction is correct and verified by an adequate number of nodes, it is considered successful. The Recipient Server extracts the verification-hash from the transaction details. The Server also calculates the hash of a string resulting from the concatenation of the sender e-mail address, recipient e-mail address and the subject. The calculated hash is compared to the verification-hash obtained from the cryptocurrency transaction details. If the two matches, the Server can be sure that the given transaction has in reality been made for the e-mail message being received. The Recipient Server also compares the transaction date with the e-mail sending date to ensure that a payment made for an e-mail is not being used again for a similar e-mail the next day. Suppose the E-mail Server cannot verify the transaction for an e-mail message, depending on the policy. In that case, the Server may mark the e-mail message as SPAM or send a bounce e-mail back to the E-mail Sender, explaining that an *'EmailSTAMP'* is required for delivering the e-mail message. The transaction verification of *'EmailSTAMP'* header in an e-mail message is shown in Fig. 8.

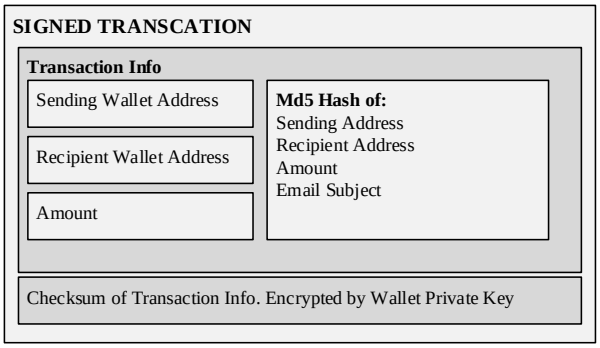

Fig. 4. A Cryptocurrency Transaction showing the Necessary Elements.

From: Alice (alice@sending-server.com)

Subject: Offer, Flat 50% of on all products.

Date: December 25, 2020 3:30:58 PM PDT

To: bob@recipient-server.com

Return-Path: <alice@sending-server.com>

Envelope-To: bob@recipient-server.com

Delivery-Date: Fri, 25 Dec 2020 15:31:01 -0700

EmailSTAMP: currency=DOGE; trx-hash=fef584b0bc69af96565cc7541gryhA0af756id8f7rtp0af6daf498ytFDNjH5faf099a

More Headers...

Fig. 5. Sample E-mail Headers Containing the Cryptocurrency Payment Information in the *EmailSTAMP*.

F. E-mail Delivery to the Recipient.

If the Recipient Server can successfully validate the transaction and confirm the payment made for the e-mail being received, it delivers the e-mail message to the user's inbox. The user can then open and read the e-mail message. The E-mail Client is modified to display the amount of cryptocurrency received for the e-mail address, after opening the e-mail message. The E-mail Client also shows the total Cryptocurrency amount available in the user's wallets.

IV. Testing and Implementation

The proposed solution has been implemented and tested using an Ethereum Test Network [23]. A Token named "EmailSTAMP" was created on the Ropsten Test Network. The Token is a standard ERC20 token with a fixed number of tokens, and the contract has only standard rules. The agreement was developed using Solidity language and deployed and tested on the Ropsten Test Network. Fig. 6 shows a screenshot of the 'EmailSTAMP'ERC20 Token on the Ropsten Test Network. The screenshot shows the Contract

Address of the 'EmailSTAMP' in the address bar and the Total Supply, which is 100 million. The 'EmailSTAMP' Token was created with support for 18 decimal places. Therefore, a tiny fraction of the Token can also be transferred in transactions.

The Smart Contract was designed such that the users or E-mail Senders can send the verification-hash along with the Token Transfer transaction. The transfer() functions of the Smart Contract accepts an additional string parameter, viz. 'verificationHash', which is meant for sending the verification-hash to the recipient for payment verification.

When calling the transfer() function on the contract, the "verificationHash' parameter should be supplied. In case the 'verificationHash' parameter is not provided, the contract execution will fail, and the entire cryptocurrency transaction will fail. Fig. 7 shows a screenshot of Remix Ethereum IDE's transfer function, showing the 'verificationHash' parameter, which accepts a string value. Fig. 9 shows the input data of a successful transaction, in which one 'EmailSTAMP' token was transferred. Note that the transaction shows 10^18 tokens equal to one 'EmailSTAMP' Token because, in the Smart Contract, the number of decimals for the Token was set to 18. This can help send a tiny fraction of the Token in a transaction instead of sending a full Token.

Fig. 9 also shows a value in the *'verificationHash'* parameter, the MD5 hash of a string created by concatenating the sender e-mail address, recipient e-mail address, amount and e-mail subject of an e-mail message.

An SMTP Server was written in JAVA and set up on two AWS EC2 Instances, one used for sending e-mail and the other was used as a Recipient. The proposed Wallet Distribution Protocol for requesting and sharing Wallet address was added to the JAVA based E-mail Server. MySQL database server was used to store wallet private keys and wallet addresses. The e-mail server was developed to create transactions on the Ethereum Test Network, create the 'EmailSTAMP' header, and add it to the outgoing e-mail. The e-mail server also verifies transactions on the blockchain, if an 'EmailSTAMP' header is found in the incoming e-mail messages.

An open-source PHP based e-mail client, *SquirrelMail*, was enhanced to work with and support the custom JAVA based SMTP Server. The enhanced E-mail Client was hosted on both the AWS EC2 instances and configured with the respective E-mail Servers. The modified E-mail Client can also create the wallet, store it in MySQL database. On the Sending Server, the Client can also use the Wallet Distribution Protocol to connect to the Recipient Server to obtain wallet address for an e-mail address. It can create a Transaction and add the transaction hash to an e-mail message *'EmailSTAMP'* header before submitting it the E-mail Server.

The E-mail Client was able to create the wallet and store it in the MySQL database. Before sending an e-mail, the E-mail Client successfully obtained the wallet address of an e-mail address on the Recipient Server. It then generated and broadcasted a transaction on the Ropsten Network, created an 'EmailSTAMP' e-mail header, added it to the e-mail and submitted the e-mail to the E-mail Server for Delivery. On the Recipient Server, the e-mail headers were obtained by the SMTP Server. The transaction was extracted, and the transaction was verified on the Ropsten Test Network after which the e-mail was successfully delivered to the recipient's inbox. Another test message, which was sent without adding an 'EmailSTAMP' header, was sent to the Recipient Server which was successfully filtered out and sent to the SPAM folder.

Fig. 6. Screenshot of the 'EmailSTAMP' *Token* on the Ropsten Test Network.

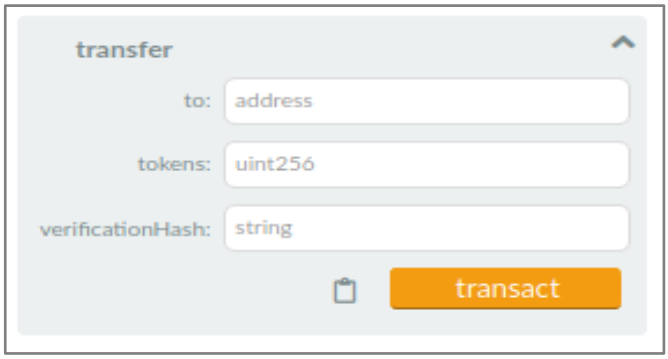

Fig. 7. Screenshot of transfer() function from Remix Ethereum IDE.

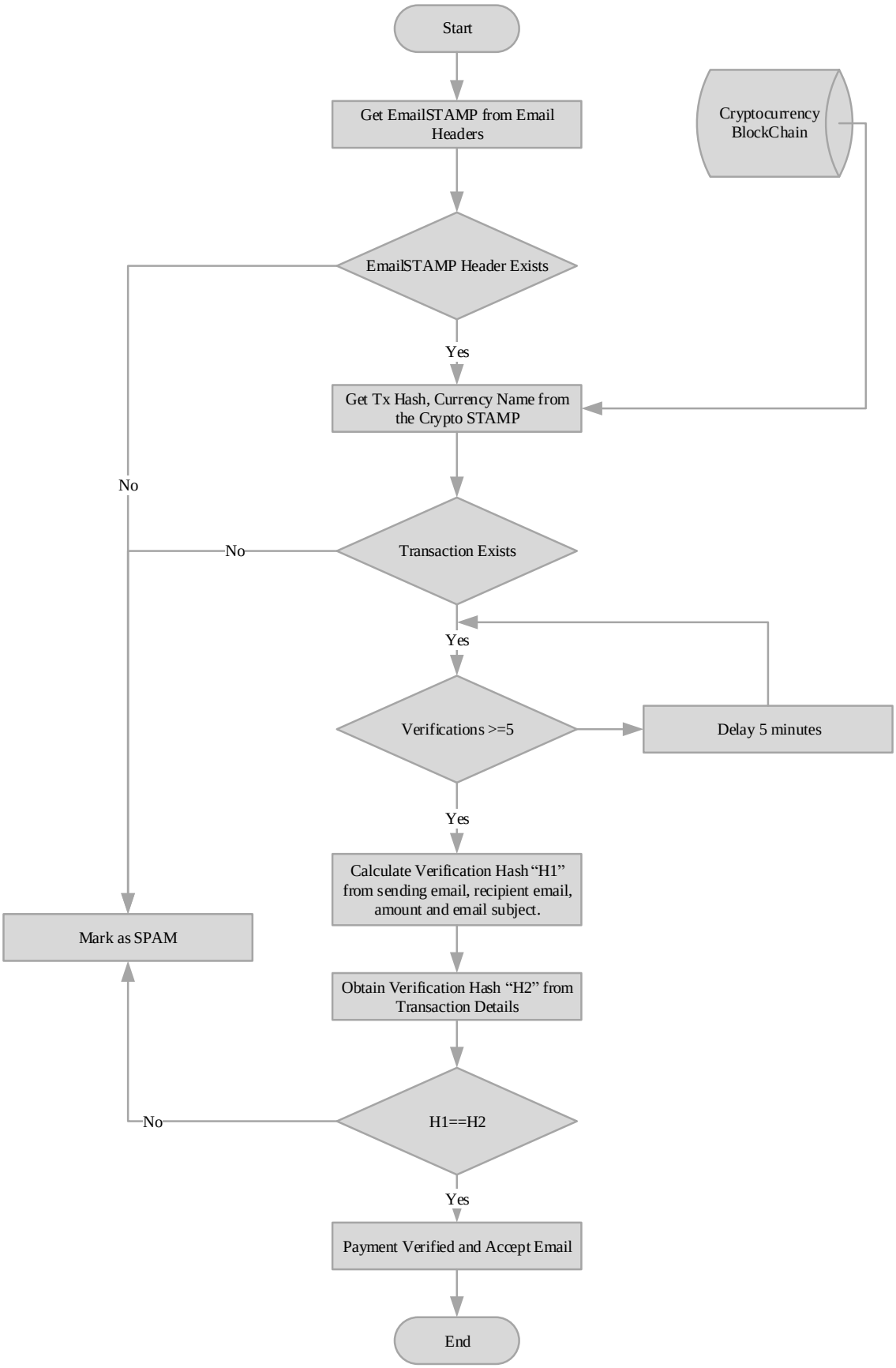

Fig. 8. Verification of the 'EmailSTAMP' *Header*.

Fig. 9. Screenshot of the 'EmailSTAMP'token on the Ropsten Test Network.

V. Limitations and Future Scope

Although a minimal fee payment per e-mail can serve the purpose of reducing e-mail SPAM, the introduction of a fee for a service which is otherwise free will be an inconvenience for ordinary e-mail users. For such users, large E-mail Service Providers can pay the fee on their behalf. The E-mail Service Providers can make the payments from the amount they collect for incoming e-mails. Therefore, the common users might not even be required to purchase any cryptocurrency at all.

The introduction of extra security measure, no doubt, will have an impact on any process. Likewise, adding a payment system into the e-mail delivery process will introduce some delay in e-mail delivery. Generating and broadcasting a transaction on an Ethereum network will take no more than a second; therefore, sending side will not notice any delays. On the other hand, Ethereum transactions take between 15 seconds and 5 minutes, to complete. Thus, the e-mail recipient will have the e-mail delivered in their inboxes as quickly as within 15 to 20 seconds or may be delayed by about 5 minutes. Time taken to complete and confirm a transaction varies for different cryptocurrency blockchain networks.

There is a need to research various currently available cryptocurrencies regarding their feasibility for implementing the proposed solution. The research can compare and contrast the multiple cryptocurrencies regarding their support for publishing the "verificationHash" and the transaction, transaction verification and processing speed, and transaction fee. The research will help in implementing the proposed solution most effectively.

VI. Conclusion

E-mail senders can be made to pay tiny amounts for sending an e-mail negligible for legitimate users but accumulates to form a considerable part for spammers sending bulk e-mail messages. Setting up a payment for such purposes can be extremely difficult using currently available payment infrastructure. It will be costly, if not impossible, for the e-mail recipient to verify if the e-mail sender has paid for incoming e-mail messages. This problem can be solved by using existing blockchains and cryptocurrencies. This will require some modifications to the E-mail Servers and E-mail Clients and implementing an additional communication protocol. In the proposed solution, the E-mail Server can make payments for the outgoing e-mail messages on behalf of its e-mail addresses or the end users can make the payments themselves using their E-mail Clients. At the recipient's end, these payments can be collected by the e-mail recipients or the E-mail Server and used to send out e-mail messages or be sold on cryptocurrency exchanges.

The work presented in this paper suggests not to create any new cryptocurrency or set up a new blockchain for processing the payments for several reasons. It is tough to use new cryptocurrencies, making it more feasible to use any existing reliable cryptocurrency. Cryptocurrencies need miners to verify transactions and support the blockchain, which is only possible if they find the process profitable. New cryptocurrencies are not attractive for miners at all and are, therefore, challenging to implement. Hence, using an existing cryptocurrency is recommended and demonstrated in this study.

The proposed system is backwards compatible because it does not attempt to modify the basic SMTP protocol or any other established e-mail communication protocols. Existing SMTP servers can receive e-mail from E-mail Senders even if they support the proposed system. In the absence of the payment header, a Recipient SMTP Server which supports the proposed solution will still accept e-mail from the Sender, but it will deliver the message to a SPAM folder instead of inbox.

References

[1] Spam: Share of Global Email Traffic 207-2019, https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/, accessed October 2020.

[2] U. H. Rao, U. Nayak, "Cryptography," An Introduction to Information Security, Springer, September 2014.

[3] E. Kirda, C. Kruegel, "Protecting Users against Phishing Attacks," The Computer Journal, vol.49, January 2006.

[4] W. Z. Khan, M. K. Khan, F. T. B. Muhaya, M. Y. Aalsalem, H.C. Chao, "A Comprehensive Study of Email Spam Botnet Detection," IEEE Communications Survey and Tutorials, vol. 4, pp. 2271-2295, June 2015.

[5] A. Karim, S.Azam, B. Shanmugam, K. Kannoorpatti, M Alazab, "A Comprehensive Survey for Intelligent Spam Email Detection," IEEE Access, vol. 7, November 2019.

[6] W. Koch, "The GNU privacy guard," http://www.gnupg.org, accessed November 2020.

[7] PGP, "Pretty Good Privacy (PGP)," http://www.openpgp.org, accessed November 2020.

[8] J. Schaad, B. Ramsdell, S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME)," RFC 5751, Version 4.0, April 2019.

[9] J. Linn, "Privacy-Enhanced Mail (PEM)," RFC1421, February 1993.

[10] J. Callas, L. Donnerhacke, IKS GmbH, H. Finney, D. Shaw, R. Thayer, "OpenPGP," RFC 4880, November 2007.

[11] D. Crocker, T. Hansen, and M. Kucherawy, "DomainKeys Identified Mail (DKIM) Signatures," RFC 6376, September 2011.

[12] Y. Yuan and F.-Y. Wang, "Blockchain and Cryptocurrencies: Model, Techniques, and Applications," *IEEE Trans. Syst. Man Cybern, Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[13] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. https://bitcoin.org/bitcoin.pdf, accessed October 2020.

[14] V.Buterin, "White Paper- Ethereum," https://ethereum.org/en/ whitepaper/, accessed November 2020.

[15] C. Lee, White Paper- Litecoin, https://litecoin.org/, accessed October 2020.

[16] S. Ellis, A. Jules, S. Nazarov, "ChainLink: A Decentralized Oracle Network," URL: https://link.smartcontract.com/whitepaper, accessed: October 2020.

[17] D. Schwartz, N. Youngs, A. Britto, "The Ripple Protocol Consensus Algorithm," https://ripple.com/files/ripple_consensus_whitepaper.pdf, accessed November 2020.

[18] D. Mazieres, "The Stellar Consensus Protocol," https://www.stellar.org/papers/stellar-consensus-protocol?locale=en. Accessed October 2020.

[19] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[20] David A. Turner and Keith W. Ross, "The Lightweight Currency Protocol (LCP)," September 2003. http://www.ietf.org/internet-drafts/draft-turner-lcp-00.txt.

[21] K. Nakayama, Y. Moriyama, C. Oshima, "An algorithm that prevents spam attacks using blockchain," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 9, 2018.

[22] S. Dennis, "Credo Token-Blockchain Based Spam & E-mail Access Solutions,", Turing Technology, Inc., July 2017.

[23] Ropsten Testnet Explorer, https://ropsten.etherscan.io/, accessed December 2020.