Discouraging Spam Attacks with Email Payment Stamps

Evan Parry, Shubham Mazumder, Yunhang Zhang

Spam Prevention

- Spam is a persistent nuisance
- Spam operations are cheap to operate
  - Most of the required resources have little to no cost
- Spammers profit from people opening emails and visiting associated links (advertising, phishing scams, etc.)
- Solution: make spamming more expensive, especially when spam emails aren’t opened

Overview

- Email server is set up so that an email must include valid “payment” in order to be sent to the client.
  - Payment could be cash, cryptocurrency, etc.; something that incurs a cost for the sender.
  - Small enough to not overly inconvenience legitimate senders, large enough to inconvenience spammers sending messages en masse
- The server initially only gives the client basic information about the email (sender, subject line, etc.).
- If the client decides to open the email, the server is automatically notified, and the payment is refunded to the sender.
- Intended result: in the long run, spammers face increased expenses, while legitimate emailers experience no net cost, thereby discouraging spam.

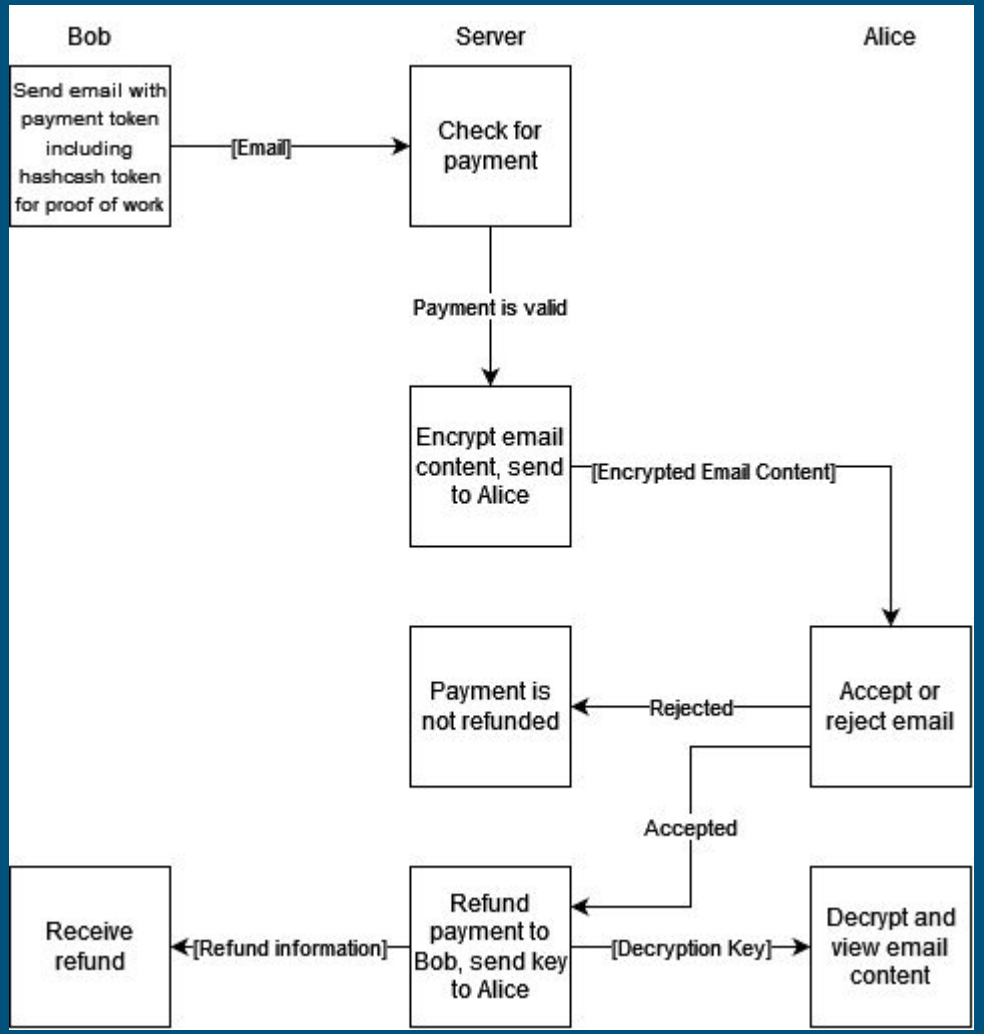
Adversary Model

- Two adversary types:
  - Spammer: capable of sending spam emails, capable of making payments
  - Receiver: capable of receiving emails, also capable of rejecting legitimate emails and not refunding them
- Defeating spammer adversary
  - Emails that aren't paid for never make it to the client.
  - Emails that are paid for aren't guaranteed to be refunded, adding element of financial risk.
- Defeating receiver adversary
  - Refunding is not under their control; it happens automatically when they open an email.
  - Server logs email and payment information for accountability purposes.

Implementation Process

- Started with basic proof of concept
  - Socket-based messaging system written in Python
- ScroogeMail: Moved on to a more advanced version
  - ScroogeCoin: Scrooge, as the Server, mints coins and gives them out to users upon registration.
  - Spam Prevention:
    - ScroogeCoins paid to Server for every email.
    - HashCash token sent to Server for every mail to show proof of work. Also helps limit DDoS attacks.
  - Protection against receiver adversary: All emails outgoing from the Server are encrypted with ephemeral keys, recipient only receives key to decrypt email content when they choose to open the email. Upon sending key to recipient, payment is refunded back to sender.

Flow Diagram



Drawbacks

- Scrooge is the central trusted authority.
- Resource intensive, need to consider trade-off.
- Lacks end-to-end encryption.
  - Needham-Schroeder can be used to exchange keys between Client and Receiver for encryption of emails.
  - Server would still doubly encrypt with an ephemeral key to protect against recipient adversary.

Proposed Further Development

- Whitelist and blacklist filtering systems
  - Whitelisted senders don’t have to pay for messages.
  - Blacklisted senders still have to pay but their messages are automatically classified as spam and are therefore unlikely to be opened.
- Message from sender is encrypted, and sender and receiver perform a key exchange; this ensures the server can’t read the email.
- Moving to a fully decentralized cryptocurrency so that the payments made need to be minted or bought by the Sender. If the coins can only be minted, it would also allow to replace HashCash as proof-of-work.

Conclusions

- Cryptocurrencies offers us a unique way to stop spam in its tracks.
- If more and more people use proof-of-work systems, such as Hashcash, then we can begin demanding that email have valid tokens attached, or mark the mail as spam.
- Combining proof-of-work systems with payment mechanisms adds an enormous overhead for the spammer but not so much for the average user.

Code and References

- Full writeup available at https://docs.google.com/document/d/1OeQvPbOv-qD-RQiZuYp1tv6MkVY79Eh3ZEvtus8ySCs/edit?usp=sharing
- Code for Python socket system implementation
  - https://github.com/Reynolds1997/CS5490EvanParryFinalProject
- Code for final modified SMTP server implementation
  - https://github.com/sansquoi/NetSecFinalProject
- “An Algorithm that Prevents SPAM attacks using Blockchain” by Koichi Nakayama, Yutaka Moriyama, and Chika Oshima https://thesai.org/Downloads/Volume9No7/Paper_29-An_Algorithm_that_Prevents_SPAM_Attacks.pdf

Questions?