Spam Control by Source Throttling Using Integer Factorization

Rochak Gupta, Vinay Kumar K., and Radhesh Mohandas

Department of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal, India
gupta.rochak@gmail.com

Abstract. Existing solutions for spam control that are limited to spam filtering at the receiver side underestimate the fact that the network bandwidth and processing time of the recipient email servers are wasted. To cut down these costs spam should be controlled before it reaches the receiving email server. In this paper, we propose a solution to control spam at the senders email server by throttling the client's CPU using integer factorization problem. Integer factorization is used to generate stamps as a proof of CPU cycles expended by the senders system for each email recipient. Cost of generating stamps is negligible when the client is sending emails to only a few recipients. However, as the number of recipients increases, the cost of generating stamps also increases which adversely affects the processing speed of the client. The server requires minimal processing time to verify stamps generated by the client.

Keywords: Integer Factorization, Source throttling, Spam, Spam control, Stamps.

1 Introduction

Techniques that control spam at the client side decrease only the costs associated with recipients. These techniques do not reduce the costs associated with network bandwidth to carry heavy load of spam and email servers to process spam emails.

Cost based spam control can be a solution to reduce the volume of spam by making the senders to pay for each email being sent. Nevertheless, forcing legitimate sender to pay money is not a good solution to achieve spam control. However, computational proof for spam fighting is an innovative solution that makes senders pay for sending email using computational effort rather than money [1], [2]. The idea is to make the sender pay some digital cost by performing a complex computation as evidence that email is worth receiving. This processing time is a minimal burden upon legitimate senders who send few emails every day. However, a spammer simply cannot afford to spend this additional time without slowing down spamming activity.

Our proposed solution is a cost based spam control. It controls spam at ingress point (sender email server) by throttling the client's CPU i.e., to make clients pay a stamp fee for each email recipient. The solution is based on integer factorization which is one of the most complex mathematical problems to solve.

K.R. Venugopal and L.M. Patnaik (Eds.): ICIP 2011, CCIS 157, pp. 233–239, 2011.

2 Related Work

The most common approaches are blacklist and whitelist. While whitelist is effective technique, it has several drawbacks. Any email sent by a stranger will be incorrectly classified as false positive (FP). The major flaw of blacklisting is that spammers tend to forge header information like sender information in spam emails and legitimate senders are also being added to blacklists.

Other spam filtering techniques are content based, phrase based and rule based. The problem with these techniques are that they need constant update and refinement because spammers use obfuscation techniques. Even some times constant updates do not work. One common obfuscation technique is using leet characters in content to disguise it from content based spam filters [3].

Researchers are working on spam filters to increase accuracy [4], [5]. Even though, the above mentioned approaches are good enough, they have two flaws. First, even if spam filters are fine tuned, they block or misclassified legitimate messages as spam (false positive) [6], [7]. The damage of a single false positive can be very serious [8]. Another problem is that they filter a message after it is delivered and stored in the receiver's email server. Cost of processing these emails at recipient email server and network bandwidth wastage are same.

3 Proposed Approach

The modified client server communication procedure with our proposed solution to achieve spam control is explained as follows:

1. At server, a list of prime numbers is generated and stored. Lists of composite numbers are generated by simply multiplying the prime numbers. e.g., $N = p * q$, where 'p' and 'q' are prime numbers and N is a composite number.
2. Sender composes an email and clicks on send. After the sender clicks on send, email client sends a 'helo' SMTP command and server replies with '250 ok'.
3. Client sends 'mail from' command and server replies with '250 ok' message.
4. For each recipient email id, client sends *rcpt to* command. Server maintains lists of composite numbers. For each session server selects one list and fetches a composite number from that list for each email recipient. Server injects the selected composite number with the reply message of *rcpt to*. Fig. 1 shows SMTP client server communication state diagram with our solution.
5. From each reply of *rcpt to* command, client extracts composite number injected by the email server. The client computes p and q using integer factorization algorithm given in section 5.1.
6. Client builds stamps using $<N, p, q>$ triplets. Following is format of stamps:

   X-INTEGERFACT: p: q: N

   Client computes stamp for each recipient of the email, and appends the generated stamps with body of the email.
7. Client forwards the email to the sender's email server. The server extracts stamps appended with email body and verifies them. The server performs verification in two steps. In first step, the server checks whether $N = p * q$.