Spam Detection and Filtering on P2P System Using Agents

Packiavathy .M
Department of Computer Science and Engineering
Pondicherry Engineering College
Pindicherry, India
packiavathy_m@yahoo.co.in

Saruladha .K
Sr. Lecturer
Department of Computer Science and Engineering
Pondicherry Engineering College
Pindicherry, India
charusanthaprasad@yahoo.com

Abstract— **Domain Keys Identified Mail (DKIM) defines a mechanism for using digital signatures on E-Mail at the domain level, allowing the receiving domain to confirm that E-Mail came from the domain it claims to. Erasure codes introduce pollution attack, an attack in which the adversary injects packets to disrupt the erasure decoding procedure and consequently denies the authentication service to the receiver. This paper uses the agent for checking the spam E-Mail by using DKIM and proposes a new lightweight, pollution-attack resistant multicast authentication scheme (PARM), which generates evidence that receiver agent can validate on a fast, per-packet basis. Using agent, the authenticity of the system will increase and also our system will be more secure. Because of using the temporal key, time will be saved for generating evidence to the same message.**

Keywords- ***DKIM, Evidence, Pollution attack***

Introduction

In [1] SpamWeeder (a simple idea for spam prevention) is discussed. The SpamWeeder approach integrates two subsystems, a track & kill subsystem that allows a user to identify roots of the spammer trees which blocks mail from entire spammer trees, and an early warning subsystem that receives input from the track & kill subsystem and warns users of the web sites involved in E-Mail address trafficking. In [2] the use of DKIM signatures and signing practices provides sending domains a tool to help recipients identify legitimate messages from their domain, and a reliable identifier that can be used to combat spam. In [3] to tolerate packet loss, erasure codes are employed to enhance signature amortization. However, the use of erasure codes introduces pollution attack, an attack in which the adversary injects packets to disrupt the erasure decoding procedure and consequently denies the authentication service to the receiver. Lightweight, pollution-attack resistant multicast authentication scheme (PARM), which generates evidence that receivers can validate on a fast, per-packet basis. This approach effectively resists pollution attacks and has better performance.

In [4] identification of messages with a knowledge based system which identifies messages with slight variation is discussed. In [5] intra-domain security is discussed. It keeps E-Mail messages in corresponding mailboxes as encrypted messages. In [6] two kinds of agents for filtering spam is discussed. Local agent for filtering spam and to learn spam's feature; Social agent for searching the same or approximate E-Mails.

A. Pollution Attack

In [8] the attacker mixes polluted chunks into the P2P distribution, degrading the quality of the rendered media at the receivers. Polluted chunks received by an unsuspecting peer not only effect that single peer, but since the peer also forwards chunks to other peers, and those peers in turn forward chunks to more peers, the polluted content can potentially spread through much of the P2P network

B. Motivation and Contribution

E-Mail systems are vulnerable to a number of security risks. In an E-Mail exchange process, receiving party cannot be sure that the E-Mail is from the actual sender. Similarly, sending party is not able to make sure that intended recipient has received E-Mail. Since E-Mail is delivered with other Internet traffic over the same transport service, it is vulnerable to eavesdropping. Also, malicious content in E-Mail content might enter user host through E-Mail client.

Spam is one of the major problems of today E-Mail systems. While many solutions have been proposed to automatically detect and filter spam, spammers are getting more and more technically sophisticated and aware of internal workings of anti-spam systems, finding ways to disguise their E-Mails to get around the different controls that can be enforced.

This paper focuses on an agent based system which uses DKIM procedure for identifying and filtering spams. This system is also made resistant against pollution attacks.

The rest of the paper is structured as follows. Section 2 explains about the proposed sender architecture for filtering the spam. Section 3 provides the details of our proposed receiver system. The last sections concluding the proposed model.

SPAM FILTERING ARCHITECTURE WITH SENDER EVIDENCE USING AGENT

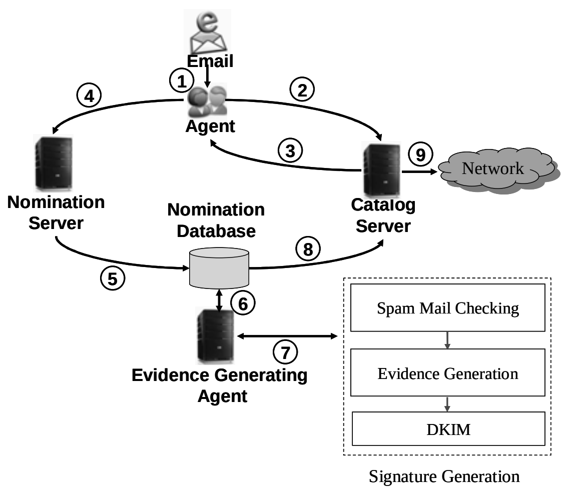

Figure 1 Architecture diagram of the sender system

Sender system sends the E-Mail to the agent. An agent residing on a user's desktop computes a DKIM signature of a new E-Mail and submits this DKIM Signature to the Spam DKIM signature Catalog server. If the Catalog server has the signature in its database, the server tells the agent that the message has been flagged by the community as spam. If the DKIM signature is not in the Catalog server, and the recipient feels that the message is spam, the recipient instructs the agent to transmit the DKIM signature to the Nomination server, which in turn, inserts the signature into the Nomination database. The Evidence Generation System continually watches the Nomination database to see if there are any new DKIM signatures that have been submitted by E-Mail system. Spam checking and the Evidence Generation with DKIM signature will be generated. DKIM signature stored in the catalog server and finally the E-Mail sends to the network.

A. Evidence Generating Agent

Once EGA (Evidence Generating Agent) determines that a signature is "spammy," then the DKIM signature is added to the catalog server. All messages received by a user are signatures and the DKIM signature are queried against the catalog server. If the queried DKIM signature exists in the catalog server, the agent filters the message as spam. If the DKIM signature is not in the catalog server and the recipient feels that the message is spam, then the recipient submits a signature to the nomination server and the process begins again.

B. SIGNATURE ALGORITHM

For generating signature EGA uses three main modules

a. Spam Mail Checking, b.Evidence Generation, c.DKIM

I. Spam Mail Checking

This module is used by EGA. Using common spam characteristics EGA can identify the spam and also control the spam. Spam characteristics are categorized into two types 1. E-Mail header and 2. E-Mail body

```
Procedure CHKJMAIL(EM)
    //EM ← E-Mail
//Assume the given E-Mail contain 'to', 'from', 'subject' and 'body' field. To,
//from, subject are considered as E-Mail Header and body contains the actual
//message
    SPAMCHAR(E-Mail_Header)//check the E-Mail Header
    SPAMCHAR(E-Mail_Body)//check the E-Mail Body
End CHKJMAIL
```

II. Evidence Generation Phase

This module is specially designed for avoiding pollution attack which is happening in P2P system. For generating evidence we are going to use two key values. Temporal Secret Key (TSK) chain and a Temporal Public Key (TPK), using a one-way hash function. The EGA creates the evidence of a packet from a TSK chain, and the receiver validates the evidence of a received packet with the TPK.

First, the EGA generates k n-bit random numbers (R_0, R_1,…, R_{k-1}) and denotes this set of numbers as TSK_0 of the TSK chain. Then, the EGA uses the oneway hash function h to recursively generate the remaining TSKs of the TSK chain. By applying the hash function to each member of the previous TSK, the EGA can produce the next TSK. The TSK chain has a length of L and is represented as (TSK_0, TSK_1,…, TSK_{L-1}). The temporal public key (TPK) is created by hashing every element of TSK_{L-1}.

```
Procedure INITIAL()
    initialize k,z,i,TSK[][],TPK[][]
//TSK – Temporal Secret Key and TPK – Temporal Public Key for storing the evidence
    R0 ← random(n) //generating the n random number
    For i ← 1 to z do
        For j ← 1 to k do
            If i==l then
                TPK[i][j] ← H(Ri)
            Else
                TSK[i][j] ← H(Rj)
                Rj ← TSK[i][j]
        End
    End
End INITIAL
```

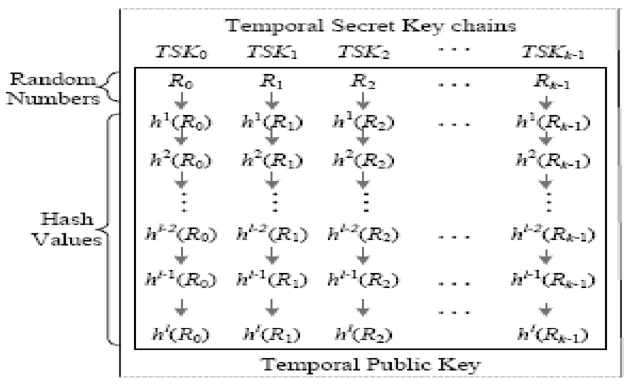
Temporal Secret Key chains
TSK_0 TSK_1 TSK_2 TSK_{k-1}
Random Numbers
Hash Values
R_0 R_1 R_2 R_{k-1}
$h^1(R_0)$ $h^1(R_1)$ $h^1(R_2)$ $h^1(R_{k-1})$
$h^2(R_0)$ $h^2(R_1)$ $h^2(R_2)$ $h^2(R_{k-1})$
$h^{l-2}(R_0)$ $h^{l-2}(R_1)$ $h^{l-2}(R_2)$ $h^{l-2}(R_{k-1})$
$h^{l-1}(R_0)$ $h^{l-1}(R_1)$ $h^{l-1}(R_2)$ $h^{l-1}(R_{k-1})$
$h^l(R_0)$ $h^l(R_1)$ $h^l(R_2)$ $h^l(R_{k-1})$
Temporal Public Key

Figure 2 Temporal Key Pair Generation

R0 denotes the randomly generated number, and the arrows specify the direction of the one-way hash function h. Thus, h(R_0) is the hash result of R0, and h2(R_0) is the hash result of h(R_0). The set of the elements in the same row comprises a TSK elements array, e.g. TSK0=(R_0, R_1,…, R_{k-1}) and TSK_1=(h(R_0), h(R_1),…, h(R_{k-1})). The elements of the last row form the TPK.

Prior to broadcasting a message, the sender must generate for each packet the evidence, or verification information, which allows receivers to determine the validity of a packet. Since each packet is augmented with evidence, the evidence generation phase should be lightweight and fast.

For a given temporal key pair, the sender needs to maintain a usage table, that tracks the number of times each column index of the TSK elements array is used. The row index denotes the column index of the TSK elements array, while the row usage tracks the number of uses of the corresponding index.

Table 1 Usage Table

Usage Table					
Chain Index	*TSK_0*	*TSK_1*	*TSK_2*	...	*TSK_{k-1}*
Usage Amount	*l-3*	*14*	*0*	...	*l-1*

To generate evidence EM for a packet M, the sender first hashes the packet with a one-way hash function h. The hash value is divided into a set of p segments, denoted S=(i_0, i_1,…, i_{p-1}), where each segment size is b-bits. Interpreted as an integer between 0 and 2b-1, each segment in the set S represents a column index of the TSK elements array. For each index i, the sender determines the TSK based upon the usage of i by selecting TSK(L-1)-ai, where ai denotes the usage of i.

The sender chooses the last TSK of the chain, TSKL-1, if i has never been used. Once the sender determines the TSK, it chooses the i-th element of the selected TSK. For example, if i_0 used L-1 TSK elements, then the sender chooses the i_0-th element of TSK_0, which is R_0. Since each segment of S corresponds to an index of the TSK elements array, the sender produces p elements, which constitutes the evidence of the packet. After appending the evidence to the packet, the sender can finally broadcast the packet to the receiver.Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

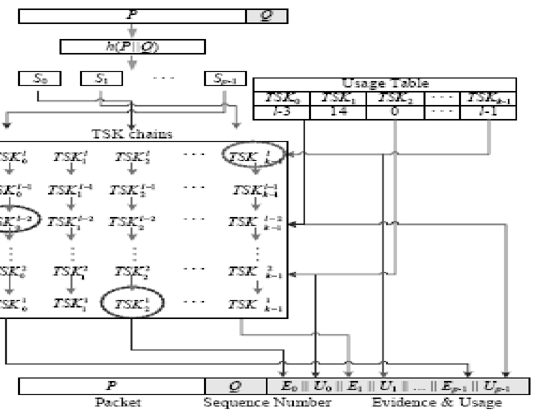
P
Q
h(P||Q)
S_0 S_1 S_{p-1}
Usage Table
TSK_0 TSK_1 TSK_2 TSK_{k-1}
l-3 14 0 l-1
TSK chains
Packet
Sequence Number
Evidence & Usage
E_0 || U_0 || E_1 || U_1 || ... || E_{p-1} || U_{p-1}

Figure 3 Evidence Generation

```
Procedure EGENERATION(P,Q)
    ← //p        packet which is going to transfer
    ← //Q        Sequence number of the packet
    H_Value ← h(P||Q)//hash and concatenates the packet with sequence
    HValue_Length ← H_Value.length()//find the length of the hashed value
    Resulted_Packet ← P||Q

    While(HValue_Length!=0)
        For i ← 1 to n do
            s[x] ← Byte[H_Value]//convert the hashed value to byte
        End
        Temp ← s[x]
        H_Value=H_Value-temp
        HValue_Length=H_Value.length()
        x++
    End //while
    Flag=true
    For i ← 1 to x do
        While(flag)
            S_Evidence ← random(TSK)
            If S_Evidence !=Content(Usage_Table)//check the evidence with //usage table
                s[i] ← s[i].append(S_Evidence)//append the evidence //in the s[i]
                Usage_table ← add(S_Evidence)//add the evidence in //the usage table
                Flag ← false
            Else
                Flag ← true
        End //while end
        Resulted_Packet=Resulted_Packet || s[i]//
    End //for loop
End EGENERATION
```

III. DKIM Signatures

Signer has decided to sign a message, it must take the following six steps

- Begin building the DKIM signature header.
- Canonicalize and hash the message.
- Select headers to be included in the signature.
- Generate a cryptographic hash of the canonical message.
- Generate a digital signature of the hash.
- Add the DKIM signature header to the message.

Signature Header

The DKIM signer must begin building the DKIM-Signature header now, since choices made through the process will be included in the header, and the header itself will be covered by the signature.

Canonicalization

It is necessary because of the long history of Internet email, the changes that have been made through that history, and the uncertainty of what a message may encounter enroute to its destination.

Select Header to be Included in the Signature

DKIM allows the signer to choose to sign some or all of the message header fields. Since many of the header fields do not contain information significant to the sender or recipient of the message, signers might choose not to sign them all.

Hash

Allows for multiple hash and signature algorithm. Hash algorithm used here is SHA-256

Signature Allow Encryption Algorithm

EGA uses two keys for generating evidence and validating the evidence. So the authenticity of system will increase and also EGA stores all the generated evidence in the nomination database and the catalog server for further checking of the E-Mail.Units.

III EVIDENCE VALIDATON FOR SPAM FILTERING USING RECEIVER AGENT

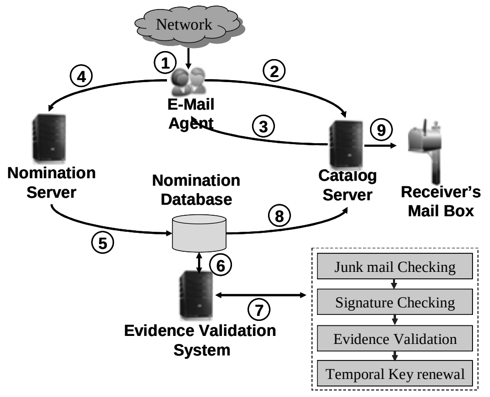

Figure 4 Block Diagram of the Receiver System

Receiver system uses the EVS (Evidence Validaton System) for validating the received E-Mail. The system receives the E-Mail from the network. An agent residing on a user's desktop in the receiver system verify the DKIM signature of a received E-Mail and also with this DKIM Signature to the Spam DKIM signature Catalog server. If the Catalog server has the signature in its database, the server tells the agent that the message has been flagged by the community as spam. If the DKIM signature is not in the Catalog server, and the recipient feels that the message is spam, the recipient instructs the agent to transmit the DKIM signature to the Nomination server, which in turn, inserts the signature into the Nomination database. The Evidence validation System continually watches the Nomination database to the received DKIM signatures. Spam checking and the Evidence Validation with DKIM signature will be verified. DKIM signature stored in the catalog server and finally the E-Mail stored in the receiver's inbox.

A. *Dkim Verification*

When a DKIM-compliant MTA receives an E-Mail message, that it decides it must verify, the message may be signed, or unsigned. The message is considered to be signed if there is a valid DKIM Signature header. The verifier must carefully check the signature header for validity.

B. Verifying A Dkim Signature

Using the contents of the i=, d=, and s= fields in the signature header, the verifier determines the desired key identity, and then uses the q= field and retrieves the key from the specified key store. For q=dns, the key is retrieved by getting DNS TXT records for "selector._domainkey.domain". The verifier must then validate the retrieved key record, and extract the public key from it.

Any failures in this process result in the signature's being declared invalid. The verifier now uses the c=, h=, and l= (if present) fields to recreate the canonical message as originally signed. Using the a= field to determine the hash and encryption algorithms, it then computes the hash on the canonical message, decrypts the signature, and compares the two resulting hash values. If they are the same, then the signature is verified. If they are not, the signature is declared invalid.

C. Checking the Signing Practices

If there is no valid signature, or if the signing identity does not match the address in the message's From header, the verifier must check the signing practices of the domain in the From address. The verifier retrieves the policy through a DNS query. The domain for the query is obtained from the From address

D. The Verifier's Decision

The verifier does with all this information – whether a signature was present or not, whether it verified or not, what the sender's signing practices say – is entirely up to the verifier. Verifiers may certainly treat messages with failed signatures as being more "suspicious" than those lacking signatures, but there are reasons for message signatures to fail that do not reflect on the legitimacy of the message.

The information can be used to help decide whether to subject the message to more scrutiny, with more or less aggressive spam filters, or to allow the message to bypass such processing. Finally, the verifier may choose, apart from the options, to convey some or all of the information to the final recipient of the message. Eventually, with a standardized mechanism to convey this information, Mail User Agents (MUA) can use this to alert the user to the trustworthiness of the message.

E. Evidence Validation

Upon receiving a packet, the receiver can use the TPK to immediately check the validity of the attached evidence. To forge a packet, the attacker must generate proper evidence for a packet, which is difficult without knowledge of the TSK chain. As with the sender, the receiver must also maintain a usage table for each column index of the TSK elements array based on received packets.

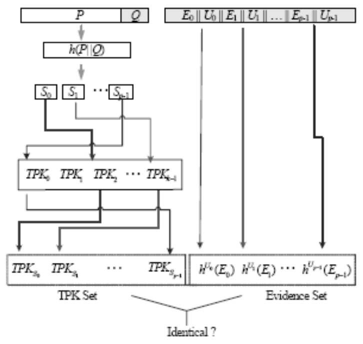
P
Q
$E_0 \| U_0 \| E_1 \| U_1 \| \ldots \| E_{p-1} \| U_{p-1}$
$h(P\|Q)$
S_0
S_1
S_{p-1}
TPK_0 TPK_1 TPK_2 $\cdots$ TPK_{k-1}
TPK_{S_0} TPK_{S_1} $\cdots$ $TPK_{S_{p-1}}$
$h^{U_0}(E_0)$ $h^{U_1}(E_1)$ $\cdots$ $h^{U_{p-1}}(E_{p-1})$
TPK Set
Evidence Set
Identical ?

Figure 5 Evidence Validation

The procedure of the evidence validation phase is similar to that of the evidence generation phase. After receiving a packet containing evidence EM, the receiver separates the evidence, denoted EM=(e_0, e_1,…, e_{p-1}), from the packet M. To validate the evidence for this packet, the receiver hashes M with the one-way hash function h, which is identical to the one-way hash function used by the sender in the evidence generation phase.

The receiver divides the hash value h(M) into p b-bit segments, denoting these segments as the set (i_0, i_1,…, i_{p-1}). By interpreting each segment as an integer between 0 and 2_{b-1}, each segment can represent a column index of the TSK elements array. Each index i, along with its usage a_i, determines the number of times to hash the corresponding element e_i of the evidence. Given an index and its usage, the receiver should perform a_{i+1} hashes on the corresponding element of the evidence.

Thus, if index i has never been used before, the receiver need only hash e_i once. The ensuing set of hash results from every element of the evidence is denoted by HR=(h_0, h_1,…, h_{p-1}). The receiver selects the verification subset VS=(hL(Ri_0), hL(Ri_1),…, hL(Ri_{p-1})) from the TPK, where hL(R_i) is the i-th element of the TPK. The receiver considers the evidence valid if the two sets, HR and VS, contain identical elements, accepting the packet with valid evidence and dropping it otherwise.

F. Temporal Key Renewal Phase

Periodic renewal of used TSK elements is necessary to ensure secure communications between the sender and its receivers. User define a threshold value T in key renewal phase. $UTSK_0$ represents the number of used elements in TSK_0 (the first TSK of the TSK chain) since the last temporal key renewal, and the set (j_0, j_1,…, j_{t-1}) denotes the indexes of the used elements. When $UTSK_0$ exceeds the threshold T, new elements are required.

The sender generates $UTSK_0$ new random numbers for the used indexes of TSK_0. Using these random numbers, the sender creates the partial TSK and the partial TPK with the one-way hash function h by following the temporal key generation procedure of the initialization phase. The sender then updates its copy of the TSK chain with the partial TSK elements. Since the receiver must also update its TPK, the sender concatenates the new partial TPK with its digital signature Sign(Partial TPK), which it then encodes with erasure codes and appends to outgoing packets. Successful renewal of the TSK chain and TPK, the sender and receiver may resume evidence generation and verification of packets.

IV. Pollution Attacks Resistant

- Attacker is unable to produce valid evidence for attack packets
- Attack packets can't pass evidence validation procedure at receiver
- Only valid packets will be accepted

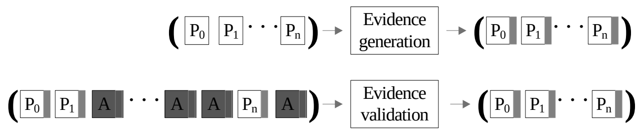

Figure 6 Pollution Attacks Resistant

Receiver agent uses the temporal key renewal for validating the threshold value. So it will resist the attacker to generate the evidence.

IV SIMULATION RESULT

The algorithms were developed and was coded using Java. For the experiments conducted a sample collection of Spams from different mail servers were collected. The spams that were identified by known mail servers were correctly identified by the system. In addition to that some spams misclassified as E-mails were identified as spam in our system. A summarization of the number of spams collected from different mail servers is tabulated in table 2.Figure 7 Shows the spam identification ratio using collected E-Mail. The graph shows Mailing system identifies the spam mail more when compared to yahoomail, gmail and hotmail.

The proposed DKIM system was tested by running a number of simulation runs. The system is tested for correct identification of spam, resistant to pollution and sybil attacks. For the testing of the system around spam were collected for different E-Mail system. These spam were simulated and was tested.

Table 2 : Number of E-Mail Tested

E-Mail	Spam
Yahoomail	400
Gmail	400
Hotmail	400

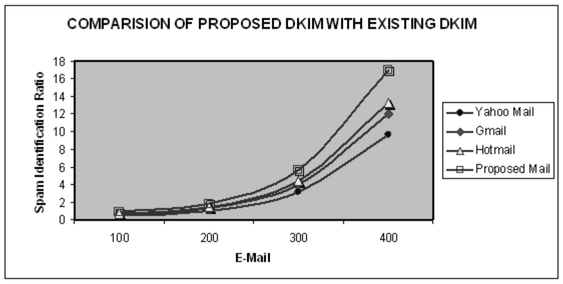

Figure 7 Spam Identification Ratio

$$\text{Spam Identification Ratio} = \frac{\text{Spam Hits/Total Spam}}{(\text{Spam Hits/Total Spam}) + 2 * (\text{Innocent Hits/Total Innocent})} \qquad (1)$$

The figure 8 shows the pollution attack ratio. In this figure X axis shows the number of evidence using and Y axis shows the pollution level. It shows that the number of evidence increased then the polluted level will decrease.

$$E^{(n)} = \frac{y^{(n)} - z^{(n)}}{y^{(n)}} \qquad (2)$$

Where,
E = pollution level
y = total number of packets
z = polluted packets

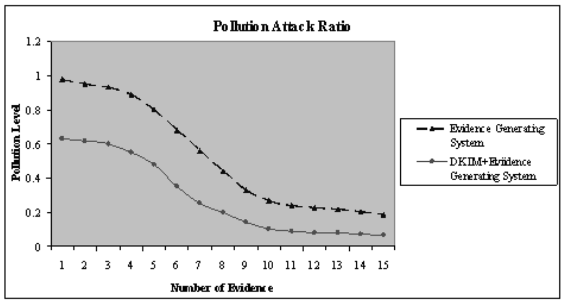

Figure 8 Pollution Attack Ratio

The figure 8 shows the security strength of the Spam Detection system. In this figure X axis shows the number of TSK Symbols using and Y axis shows the Security Strength of the system. It shows that the number of TSK element increased then the security strength of the system will also increase.

$$S = \left[\frac{Lk}{n}\right]^{P} \qquad (3)$$

Where,
S = Security Strength
L = TSK hash chain length
K = Element per TSK
P = Number of element in each evidence

N = Number of TSK in each packet

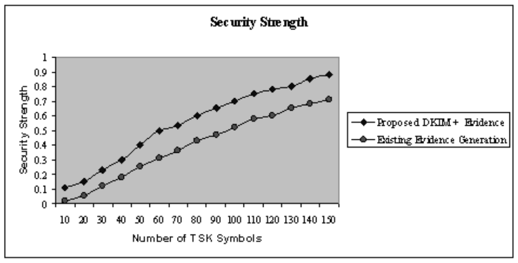

Figure 9 Security Strength

V. CONCLUSION

Spam is considered as a serious problem since it causes huge losses to the organization due to bandwidth consumption, mail server processing load, and user's productivity. The objective of work is to design a SpamWeeder for a P2P system which secures E-Mail server from receiving and sending Spams. This paper gives the idea of fighting spam E-Mail, which allows users to precisely expose parties engaged in E-Mail address trafficking and block all E-Mail from particular party belonging to a given trafficking chain. The presented trusted E-Mail system where E-Mail messages are ensured to be protected from any unauthorized access and misuse from malicious mail intermediaries can be achieved by continuously checking the E-Mail and Spam characteristic.

The ability to spoof the origin addresses of messages is a design characteristic of Internet mail that has legitimate as well as illegitimate uses. Systems that authenticate E-Mail messages must therefore be flexible enough to accommodate legitimate uses of spoofing. DKIM is designed with these characteristics. Hence a system is designed for weeding out Spams from a P2P network. DomainKeys Identified Mail (DKIM) defines a mechanism for using digital signatures on E-Mail at the domain level, allowing the receiving domain to confirm that E-Mail came from the domain it claims to. This paper also modify the DKIM by encrypting the content of the E-Mail.

Erasure codes introduce pollution attack, an attack in which the adversary injects packets to disrupt the erasure decoding procedure and consequently denies the authentication service to the receiver. This paper propose a new lightweight, pollution-attack resistant multicast authentication scheme (PARM), which generates evidence that receivers can validate on a fast, per-packet basis. and provides a solution to resist pollution attack which offers lightweight computational overhead to the sender and receiver. It also allows the receiver to instantly validate packets. The partial key renewal mechanism provides a guarantee on a lower bound of the security regardless of the amount of disclosed TSK elements.

In this work, the design of the proposed system which consists of the controlling pollution attack and trusted E-Mail delivery between two parties by continuously checking the Spam characteristic has been completed. Pollution attack is a significant problem in multicast authentication. This project proposes a new approach to resisting pollution attack that not only offers lightweight computational overhead to the sender and receiver and also allows the receiver to instantly validate packets without the need to buffer invalid packets. The partial key renewal mechanism provides a guarantee on a lower bound of the security regardless of the amount of disclosed TSK elements has been completed. The experiments show that the proposed DKIM system is secure and robust against pollution attacks.

REFERENCES

[1] Tu Ouyang and Michael Rabinovich, "Weeding Spammers at the Root : A Precise Approach to Spam Reduction," IEEE Transactions on EECS Department, 2008.

[2] Barry Leiba, Jim Fenton, "DomainKeys Identified Mail (DKIM) UsingDigital Signatures for Domain Verification," CEAS 2007 – Fourth Conference on Email and Anti-Spam, 2007.

[3] Ya-Jeng Lin, Shiuhpyng Shieh, Warren W. Lin, "Lightweight, Pollution - Attack Resistant Multicast Authentication Scheme," ASIACCS'06, March 21–24, 2006.

[4] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, *"P2P-Based Collaborative Spam Detection and Filtering,"* Fifth International Conference on Information Security, page(s):163 – 169, October 2006.

[5] Erkut Sinan Ayla Havelsan,Ankara,Attila Ozgit, *"An Architecture for End-to-End and Inter-Domain Trusted Mail Delivery Service,"* IEEE International Symposium on Computer Networks, 2006.

[6] Haixia Cao, Jianshe Dong, *"Multi-Agent Interaction Based Collaborative P2P System for Fighting Spam,"* IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2006.

[7] Brian Neil Levine, Clay Shields, N. Boris Margolin, *"A Survey of Solutions to the Sybil Attack,"* International Symposium on Communications and Information Technologies, page(s):1213 – 1218, October 2007.

[8] Julian Jang, Surya Nepal, John Zic, *"Trusted Email Protocol: Dealing with Privacy Concerns from Malicious Email Intermediaries,"* 8th IEEE International Conference on Computer and Information Technology, July 2008.

[9] Prithula Dhungel, Xiaojun Hei, Keith W. Ross, and Nitesh Saxena, *"The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses,"* Proceedings of the workshop on Peer-to-peer streaming, page(s): 323-328, 2007.

[10] Jian Liang, Rakesh Kumar, Yongjian Xi and Keith W. Ross, *"Pollution in P2P File Sharing Systems,"* 24th Annual Joint Conference of the IEEE Computer and Communications Societies, March 2005.

[11] http://www.wikipedia.com

[12] http://www.ieeexplore.ieee.org

[13] http://www.bb-warez.com

[14] http://msdn.microsoft.com/msdnmag/issues/01/02/netpeers/print.asp

[15] http://www.microsoft.com/technet/prodtechnol/winxppro/deploy/p2pintro.mspx?pf=true

[16] http://citeseer.nj.nec.com/pujol02extracting.html