World Applied Sciences Journal 29 (Computer Sciences, Engineering and Its Applications): 181-185, 2014
ISSN 1818-4952
© IDOSI Publications, 2014
DOI: 10.5829/idosi.wasj.2014.29.csea.2258

A Novel Approach Towards Prevention of Spam, Phising, Controlling Hierarchical Access Using Domain Keys

V. Khanaa, K.P. Thooyamani and R. Udayakumar

School of Computing Science, Bharath University, Chennai-73, India

Abstract: Domain Keys are the potential way to cryptographically sign and verify the messages send by a party. Authentication of the message can be done easily if an agreed up on procedure can be followed by the communicating domains. Access controls can be granted and withdrawn by verifier of the service providing party when the party in need of service introduces a message to do so which is duly signed by the authenticated and negotiated signer. The signer can negotiate with the verifier using the signature suites like **RSA-SHA1, RSA-SHA256 and Diffie-Hellman-MD5** and so on. The signer also encrypts the entire message with an encryption algorithm. The key used here forms the Domain Key. This is an extension to DKIM. The verifier also needs to negotiate with signer for **Canonicalization** of the message hence they would follow similar standards. The algorithms can be used for processing either header of the request or the entire request message. Spam is the unsolicited bulk e mails send by the party who needs to block the server of another party. When two parties come in to negotiation it's easy to avoid spam between them. When data are to be transferred it can be signed by the domain that needs to send the data and the verifier in the receiving domain has the verifier which uses the agreed up on domain key to finish the verifying process and the data can be render ed secure with both integrity and confidentiality.

Key words: Canonicalization · Signer · Verifier · RSA · SHA

INTRODUCTION

When users of one domain needs to communicate with the users of other domain securely, needs to share key with them but it is tedious to manage the keys of all the users. Hence we should use a better technique which is "DOMAIN KEYS". Users may send the message from each user of the domain to the administrator of the domain which may take care of the signing process and verifier can also be integrated to verify the signature when the user of the domain receives any message. Most of the service providers nowadays avail some of the fruitful services to those parties that deserve for the services but to identify whether the party is authorized to avail the service and the level to which they deserve to avail it are much difficult to manage. Hence the above said scheme may be extended to make the signer decide the access level then this can be appended to the message and sent. This can be used by the service providing party to subscribe the party for a particular level of service access.

- This scheme can be used in mailing system to avoid spam and other attacks.
- It also prevents spoofing of any type. The most common spoofing is modification of the domain
- Makes easy to transfer data between negotiated domains.

Example: User from a domain which is not negotiated and authenticated may try to introduce a message, to do so he can modify his domain name to some other domain with which the receiving domain may have had negotiated. This can be prevented the proposed scheme.

- This can be also used in scenarios where all people cannot have access to all data

Domain Keys: DomainKeys Identified Mail (DKIM) defines a mechanism by which email messages can be cryptographically signed, permitting a signing domain to claim responsibility for the introduction of a message into

Correponding Author: V. Khanaa, School of Computing Science, Bharath University, Chennai-73, India.

the mail stream. Message recipients can verify the signature by querying the signer's domain directly to retrieve the appropriate public key and thereby confirm that the message was attested to by a party in possession of the private key for the signing domain.

Comparison with Other Approaches: The approach taken by DKIM differs from previous approaches to message signing (e.g., Secure/Multipurpose Internet Mail Extensions)

- The message signature is written as a message header field so that neither human recipients nor existing MUA (Mail User Agent) software is confused by signature-related content appearing in the message body;
- There is no dependency on public and private key pairs being issued by well-known, trusted certificate authorities;
- There is no dependency on the deployment of any new Internet protocols or services for public key distribution or revocation;
- Signature verification failure does not force rejection of the message;
- No attempt is made to include encryption as part of the mechanism;
- Message archiving is not a design goal.

DKIM:

- Is compatible with the existing email infrastructure and transparent to the fullest extent possible;
- Requires minimal new infrastructure;
- Can be implemented independently of clients in order to reduce deployment time;
- Can be deployed incrementally;
- Allows delegation of signing to third parties.

Signing: To sign a message we may employ a signer who would use one of the standard algorithms to sign the message and forwards it to the verifier. These may be MUAs (Mail User Agents), MSAs (Mail Submission Agents), MTAs (Mail Transfer Agents), or other agents such as mailing list exploders. In general, any signer will be involved in the injection of a message into the message system in some way. The key issue is that a message must be signed before it leaves the administrative domain of the signer.

Verifier: When such a message signed by the appropriate signer reaches the receiving domains' administrator they employ their verifier to match the encrypted signature with the signature it finds by making use of the received message. If successful authentication takes place, then the message would be send to the final destined user. If it is a request for a service then the particular service can be provided based on the level of its accessibility. The verifier can also be any of the agents like signer.

The Rsa-sha1 Signing Algorithm: The rsa-sha1 Signing Algorithm computes a message hash SHA-1 as the hash algorithm. That hash is then signed by the signer using the RSA algorithm as the crypt-algorithm and the signer's private key.

The Rsa-sha256 Signing Algorithm: The rsa-sha256 Signing Algorithm computes a message hash using SHA-256 as the hash-algorithm. That hash is then signed by the signer using the RSA algorithm as the crypt-algorithm and the signer's private key.

Authentication of Individual User: The approval of the individual user is not delayed till verification the signer takes care of the authentication. Administrator can use any system like what the user knows, has or any other scheme. Simple password verification can be used, but transferring the password as plain text may ease the job of intrusion a mechanism can be used to encrypt the password by the user interface and then sent to the administrator which takes care of the either decrypting and comparing or comparing with the stored encrypted password. The algorithms used for the authentication may be RSA, Data Encryption Standards or even simple transposition cipher depending on the threads of the particular domain. Once the user gets authentication it can communicate with any domain.

Protocol:

- The domain which needs to communicate with the other domain first asks for the negotiation from its administrator
- When the other domain also needs the communication it accepts to the negotiation by sending the public key that should used by the other domain when it receives any message and requests for its public key

- Then the other domain responds to its request
- Both the domain stores the keys may be in encrypted format if necessary.
- They also share the signature suite that is to be used between them.
- Whenever any user of the negotiated domains needs to communicate with the users of the other domain first, the individual user is authenticated by the user's domain itself.
- Signer receives the message of its user and finds the hash function to that message and then uses the private key to encrypt the message and attaches it to the message and forwards to the other domain
- The other domain on the reception of the message uses the stored public key to decrypt the attachment and extracts the hash.
- It uses the hash algorithm to find the hash of the message.
- If the two hashes meets then the message is not been modified and free f4rom any active attacks.
- In the case of service subscription the signer appends the level or the verifier posses the levels already got from the signer to access the message.
- Receiving domains verifier checks the level of access and makes an entry in the data store.
- Whenever the administrator decides to send a message in connection with the service does so by specifying the level of the customer to which the message needs to reach
- Signer finds all the eligible users and forwards the service message
- In case of transfers the entire data is used for hash computation and the hash is attached with the message and then the entire message is encrypted.

Thus the above said protocol allows for a secure communication with hierarchical access.

Negotiation: There should common scheme followed for negotiation with the other domains. The negotiation request should contain the requesting domain the algorithm to used for encryption, the domain to which the request is forwarded and also the algorithm used for finding the digest

From: signer@xx
To: Verifier@yy
Key:key
Hash Algorithm: SHA1/SHA256/MD5
Encryption: RSA/DES/AES

Message Leaving a Signer: It should contain the sender, the receiver, the algorithm used for the encryption, the encrypted hash and finally the message with the subject and the body of the message

Header:
From: User@xx
To: User@yy
Encrypted Hash: -----------
Hash Algorithm: SHA256
Body:
Body Subject: (In case requesting for subscription attach the level)
Body:
END STRING

Replay Attacks: In this attack, a spammer sends a message to be spammed to accomplice, which results in the message being signed by the originating MTA. The accomplice resends the message, including the original signature, to a large number of recipients, possibly by sending the message to many compromised machines that act as MTAs. The messages, not having been modified by the accomplice, have valid signatures.

Partial solutions to this problem involve the use of reputation services to convey the fact that the specific email address is being used for spam and that messages from that signer are likely to be spam. This requires a real-time detection mechanism in order to react quickly enough. However, such measures might be prone to abuse, if for example an attacker resent a large number of messages received from a victim in order to make them appear to be a spammer.

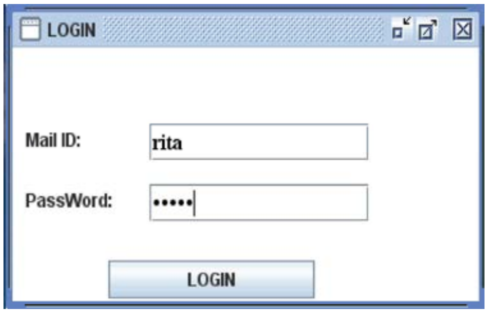
Large verifiers might be able to detect unusually large volumes of mails with the same signature in a short time period. Smaller verifiers can get substantially the same volume of information via existing collaborative systems.

Limits on Revoking Keys: When a large domain detects undesirable behavior on the part of one of its users, it might wish to revoke the key used to sign that user's messages in order to disavow responsibility for messages that have not yet been verified or that are the subject of a replay attack. However, the ability of the domain to do so can be limited if the same key, for scalability reasons, is used to sign message for many other users. Mechanisms for explicitly revoking keys on a per-address basis have been proposed but require further study as to their utility and the DNS load they represent.

Intentionally Malformed Key Records: It is possible for an attacker to publish key records in DNS that are intentionally malformed, with the intent of causing a denial-of-service attack on a non-robust verifier implementation. The attacker could then cause a verifier to read the malformed key record by sending a message to one of its users referencing the malformed record in a (not necessarily valid) signature. Verifiers must thoroughly verify all key records retrieved from the DNS and be robust against intentionally as well as unintentionally malformed key records.

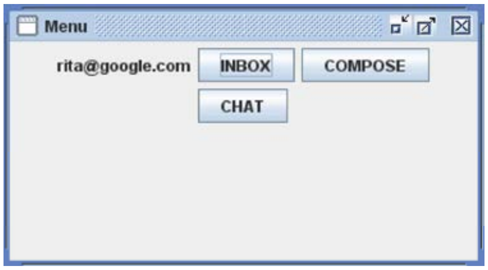
Simulation: The above scheme is used to develop a mailing system as a simulation which functions as follows.

User Authentication:



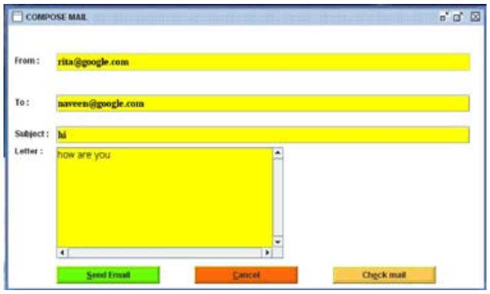
Mail ID: rita

PassWord: •••••



This window receives the user name password and sends that to the signer which checks for the correctness of the password and if it is correct accepts it

Menu:



From : rita@google.com

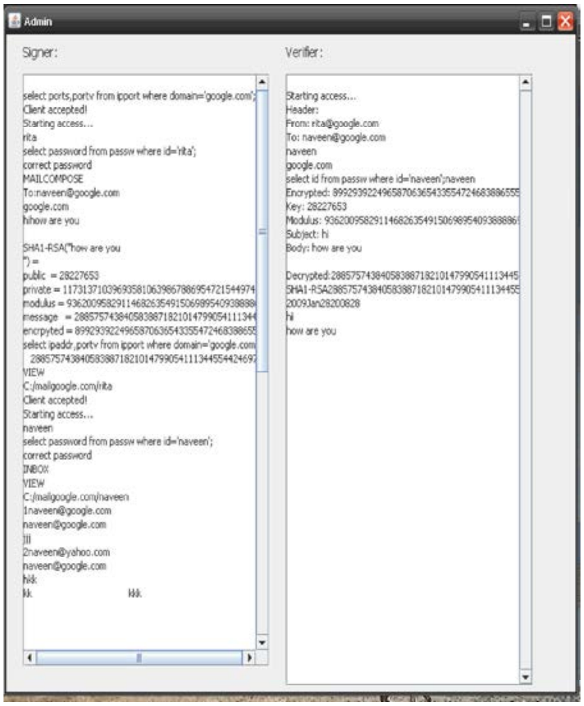
To : naveen@google.com

Subject : hi

Letter : how are you

Compose Mail: The signer and verifier of a domain are integerated

Inbox:



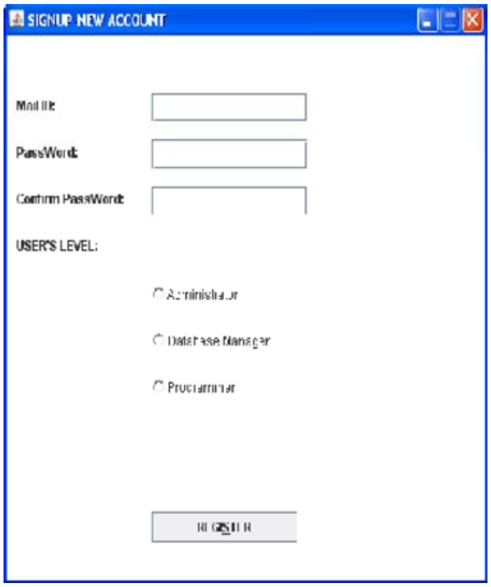
Signer:

```
select ports,portv from ipport where domain='google.com';
Client accepted!
Starting access...
rita
select password from passw where id='rita';
correct password
MAILCOMPOSE
To:naveen@google.com
google.com
hihow are you

SHA1-RSA("how are you
") =
public  = 28227653
private = 117313710396935810639867886954721544974
modulus = 936200958291146826354915069895409388886
message  = 28857574384058388718210147990541113446
encrpyted = 89929392249658706365433554724683886555
select ipaddr,portv from ipport where domain='google.com
  288575743840583887182101479905411134455442469
VIEW
C:/mailgoogle.com/rita
Client accepted!
Starting access...
naveen
select password from passw where id='naveen';
correct password
INBOX
VIEW
C:/mailgoogle.com/naveen
1naveen@google.com
naveen@google.com
lll
2naveen@yahoo.com
naveen@google.com
hkk
kk                    kkk
```

Verifier:

```
Starting access...
Header:
From: rita@google.com
To: naveen@google.com
naveen
google.com
select id from passw where id='naveen';naveen
Encrypted: 89929392249658706365433554724683886555
Key: 28227653
Modulus: 9362009582911468263549150698954093888865
Subject: hi
Body: how are you

Decrypted:288575743840583887182101479905411134455
SHA1-RSA288575743840583887182101479905411134455
2009Jan28200828
hi
how are you
```

Signer & Verifier:



Mail Id:

PassWord:

Confirm PassWord:

USER'S LEVEL:

○ Administrator

○ Database Manager

○ Programmer

Conclusion and Future Work: This scheme proves out be robust and users are handled as individual threads. This also avoids too much network traffic when multiple adjacent signers are developed and becomes the most secure system. In the future a chat application with multilevel access is to be developed in which users could communicate based on the access levels and uses this simulation as the prototype

REFERENCES

1. Crocker, D., T. Hansen and M. Kucherawy, 2010. Domain Keys Identified Mail (DKIM) Signatures.
2. Packiavathy, M. and K. Saruladha, 2009. Spam detection and filtering on P2P system using agents.
3. Higashikado, Y., T. Izu and M. Takenaka, 2008. An Extension of the Sender Domain Authentication DKIM.
4. Barry Leiba and Jim Fenton, 2007. DKIM: Using Digital Signatures for Domain Verification, CEAS, pp: 530-538.
5. Ayla, E.S. and A. Ozgit, An architecture for end-to-end and inter-domain trusted mail delivery service.