# An Add-on End-to-end Secure Email Solution in Mobile Communications

Shushan Zhao
Division of Management and Education
University of Pittsburgh at Bradford
shushanz@pitt.edu

Shuping Liu
Department of Electrical Engineering
University of Southern California
lius@usc.edu

## ABSTRACT

Many attacks, scams, and malware threats are based on or spread through emails nowadays. Although people have been fighting against them with technical and legal measures for many years, the situation is not mitigated but seems getting worse and worse, especially in the era of mobile computing. We attribute this to lack of end-to-end security solutions for emails in current Internet infrastructure. In this paper, we propose an add-on end-to-end solution for email security. It is based on idea of trust chain from sender to receiver, which spans multiple domains and organizations without requirement of a uniform platform. The solution is add-on, which means that it can be implemented on top of existing protocols as an optional component, without replacing email servers and routers in the Internet infrastructure. The solution provides end-to-end authentication and integrity for its users, which is hard to achieve in existing works.

## KEYWORDS

email, security

**ACM Reference format:**
Shushan Zhao and Shuping Liu. 2017. An Add-on End-to-end Secure Email Solution in Mobile Communications. In *Proceedings of 10th EAI International Conference on Mobile Multimedia Communications, Chongqing, China, July 2017 (MOBIMEDIA 2017),* 5 pages.
DOI: 10.475/123_4

## 1 INTRODUCTION

Nowadays, we cannot live or work without emails. Pervasion of smart phones and mobile computing devices, 4G high-speed data networks and Wi-Fi hotspots make people rely more on emails. Many attacks, scams, and malware threats are based on or spread through emails nowadays. Spamming emails and phishing emails are widely existing and serious problems, especially in mobile communications era. Some emails trick people into giving out their credentials. For example, a few weeks ago, one of the authors received an email seemingly from the president of the university, with content saying "Here is an important document all staff has to look at. It's about school updates activities. Everyone needs to read the important information carefully." Attached in the email is a pdf file. If you open the pdf file, it says "This is a confidential document, verify your identity first" and asks for user ID and password before reading it. This is an example of phishing email. Some emails with spoofed sender addresses are used to issue false messages. For example, In October 2013, an email that looked like it was from Fingerprint Cards, a Swedish biometrics company, was sent to a news agency, saying that Samsung offered to purchase the company. The news spread and the stock exchange rate surged by 50%. It was later discovered the email was a fake [8]. Some other emails pretended to be from trustable senders are used to spread viruses, Trojan horses, and other malware.

These emails are mostly from anonymous or impersonated addresses. Simple Mail Transfer Protocol (SMTP) specifies the process and requirements of sending an email. The email recipient sees the email as having come from the address in the From: header; they may sometimes be able to find the MAIL FROM address; and if they reply to the email it will go to either the address presented in the From: or Reply-to: header — but none of these addresses are required to be checked [5, 9]. Sending such emails is pretty easy. For example, recently the authors tested an email with a spoofed sender address *webmaster@example.com* sent to their email box at Vanier College. The email was successfully received, seemingly from the spoofed sender address, as is shown in Figure 1(a). The PHP script used to send the email is shown in Figure 1(b).

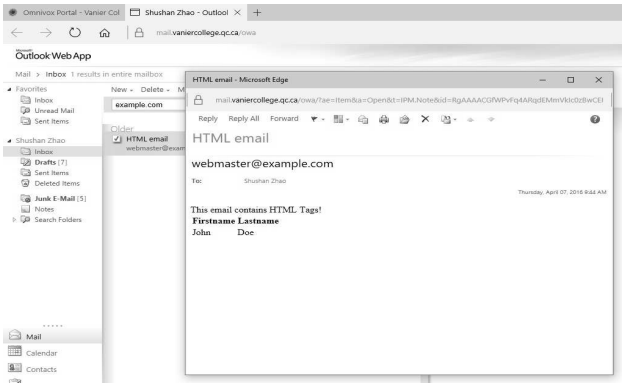
In many mail servers, the default configuration is *open mail relay.* An open mail relay is an SMTP server configured in such a way that it allows anyone on the Internet to send e-mail through it, not just mail destined to or originating from known users. The rationale of this design is: At beginning of Internet and email, only a few people had access to Internet and even fewer people had email addresses, so it was common that many people were sharing an email address. This design enabled people to send email from any place they had Internet access, and the unchecked FROM address enabled them to claim who they were or to which organization they belonged to, and to receive replies at that address. Obviously, things have changed so much ever since then. Nowadays, each email sender has her/his own address;


*MOBIMEDIA 2017, Chongqing, China*

DOI: 10.475/123_4



DOI 10.4108/eai.13-7-2017.2270117




(a) Email with spoofed sender address


(b) PHP script to send the email

**Figure 1: Email with spoofed sender address and PHP script to send the email**

sharing an email address or sending an email from a different place is not a decent requirement any more. On the contrary, still having this enabled is the major concern now. Having realized this fact, many Internet service providers block mail sent from open relay servers. This reduced the percentage of mail senders that were open relays. However, spammers and attackers have created distributed botnets of zombie computers that contain malware with mail relaying capability.

Some technical solutions have been proposed to fight against email spoofing and spamming, such as *Domain Keys Identified Mail* (DKIM), *Sender Policy Framework* (SPF), and *Domain-based Message Authentication, Reporting and Conformance* (DMARC), that we are going to review in Section 2. However, they are not end-to-end. When sending an email from universityA.com domain, if user A writes FROM: A@universityB.com, DKIM/SPF/DMARC can detect domain name spoofing; but if the sender writes B@ universityA.com, then those solutions cannot find anything wrong with it. End-to-end authentication is difficult to achieve. That is because there is no security infrastructure from sender's end to receiver's end. Theoretically Public Key Infrastructure (PKI), or personal certificates, can solve this problem; but so far PKI is only applied on domain level, and not on individual user level. For the latter, too much burden would be added to current Internet infrastructure, as each user needs to have a certificate. In summary, there are techniques to authenticate sender, to ensure sender domain, to verify email integrity en route, but they are not providing end-to-end ensurance for both the sender and receiver. By combining them, it is possible to have some end-to-end security solution, but the connecting points may be subject to man-in-the-middle attack or record-and-replay attack. In light of status quo, we propose a seamless solution to ensure end-to-end email authentication and integrity by which the receiver can verify if the sender is authentic, and the sender can verify if the receiver is authentic, and if the email content is forged or tampered en route.

The rest of this paper is organized as follows: Section 2 briefly reviews related work this work is based on. Section 3 presents the solution from overview to details. Section 4 dicusses and analyzes the security features and limitations of the scheme. Section 5 concludes the paper.

## 2 BACKGROUND

To fight against emails sent from spoofed addresses, a number of effective systems are now widely used to enforce email authentication.

DKIM allows the receiver to check that an email claimed to come from a specific domain was indeed authorized by the owner of that domain. The sender's email server signs the email with its private key if it is really from its domain, and sends the signature with the email to the receiver's server to verify. The public key of sender's server is published on Domain Name System (DNS) records of the sender's domain [3]. The purpose of a DKIM-signature is not to assure message integrity. Often, it does not even guarantee that a message author's data, as per a signed From: field, has a real name or a valid mailbox. The parts to be signed are chosen so as to identify the message unequivocally. A valid signature just states that the message did actually flow through a box operated by the sender's domain [1].

SPF provides a mechanism to allow receiving mail exchangers to check that incoming mail from a domain comes from a host authorized by that domain's administrators. The list of authorized sending hosts for a domain is published in the DNS records for that domain in the form of a specially formatted TXT record. SPF checks that the IP address of the sending server is authorized by the owner of the domain that appears in the SMTP MAIL FROM command [4]. This only ensures the email sender is indeed in the sender's domain. After this authentication, it's still possible for attackers to replace sender address, or tamper the email content, and the receiver has no means to detect that.

DMARC defines a policy that allows a sender's domain to indicate that their emails are protected by SPF and/or DKIM, and tells a receiver what to do if neither of those authentication methods passes — such as junk or reject the

message. It enables receivers to provide authentication reporting to senders to improve and monitor their authentication infrastructures [6]. DMARC is based on combination of DKIM and SPF, and it does not address the limitation of the two protocols mentioned above.

In the literature, there are also a number of works related to this topic. In [7], the authors realize that the SSL/TLS system used to encrypt server-to-server email traffic can also be used to enforce email authentication, and most of existing techniques are server-oriented and transparent to the user. They propose an SSL-based anti-spoofing application that allows a client to send trusted emails and authenticate received emails using the SSL protocol. However, it uses a self-signed certificate to exchange a secure authentication message alongside the email with a view to prevent spoofing. Self-signed certificates can only protect integrity, i.e. using the public key of the sender and signature in the email, the receiver can verify if the email is modified or not. There is no way to ensure authenticity of the sender, i.e. the sender can generate a self-signed certificate by him/herself to claim to be somebody else.

In [10], the authors demonstrate an example of email sender address spoofing by TELNET, which is from spoofed address uso@uso.uso to an authentic user at gmail.com. They propose that the receiver server obtains a sender domain name by using Auto Whois from IP address described "Received" of the most below header field, and then the sender domain name obtained by using Auto Whois, domain name below @ in the mail address written in "Received:", and "From:" are compared to detect spoofing. Again, this is only domain level authentication, and not indivisual user level authentication.

## 3 AN END-TO-END SOLUTION

The basic idea of this solution is chain of trust, i.e.: Sender's email server verifies that the sender is authentic, and signs it; the receiver's email server verifies that the sender's email server is authentic and trustable, and its signature is authentic; the receiver's email server authenticates the receiver, and the receiver trusts her/his own email server. Based on the above three conditions, the receiver ensures the sender is authentic,and the sender ensures the receiver is authentic. The logic is sound and reasonable.

Figure 2 shows the most common ways that an email message can get transferred from its author to its recipient. In this figure, a message submission agent (MSA) or mail submission agent is a computer program or software agent that receives electronic mail messages from a mail user agent (MUA) and cooperates with a mail transfer agent (MTA) for delivery of the mail. An MUA, mostly known as an email client, is a computer program in the category of groupware environments used to access and manage a user's email. A mail delivery agent or message delivery agent (MDA) is a computer software component that is responsible for the delivery of email messages to a local recipient's mailbox [2]. Our end-to-end solution is from MUA to MDA.

The process of sending an email is as follows, and an email header and body with signature components are demonstrated in Figure 3:

(1) User logs into an MUA, e.g. from a webmail page or a mobile terminal application. A password $p$ for sending email is required for this step. This password can be set to the same as the one for receiving emails by default.
(2) User inputs all header fields and body content of the email. The following steps are transparent to the user.
(3) MUA application calculates hash value of user's password. We can use the same mechanism used in Linux systems:
  (a) MUA chooses a salt value $s$ that is a random data generated to combine with the original password, in order to increase the strength of the hash.
  (b) MUA chooses a hash algorithm $H$ of index $i$ from a pre-defined list, e.g.:1 for MD5, 2 for Blowfish, 3 for SHA-256, etc.
  (c) MUA calculates a hash value $h = H(p + s)$.
(4) MSA application appends $[s_1 = \$i\$s\$h]$ to the end of message body. For example, $[s_1 = \$1\$Etg2ExUZ\$Ds5e24NuQTP2tQ8vLn5Mw]$ means: using $MD5$ hash algorithm, $Etg2ExUZ$ as salt, hash value of $H(p + s)$ is $Ds5e24NuQTP2tQ8vLn5Mw$ ($p$'s value is "*emailpswd*" in this example).
(5) MSA uses $h$ as key, and uses a predefined Message Authentication Code (MAC) algorithm, such as $HMAC$, to calculate a message tag $s_2$ of the message header $mh$ and message body $mb$ and the above $s_1$: $s_2 = HMAC_h(mh + mb + s_1)$.
(6) MSA appends $s_2$ to the end of message body, and sends it to MTA.
(7) MTA first verifies authenticity of the claimed sender in email header:
  (a) MTA derives $i$, $s$, and $h$ from $s_1$, and a hash algorithm $H$ of index $i$ from the same pre-defined list embedded in the code.
  (b) MTA retrieves sending password $p$ of the sender.
  (c) MTA calculates a hash value $h' = H(p + s)$, and compares it with the one $h$ derived from $s1$ in the message: $h' \stackrel{?}{=} h$. If it is same, then continue; otherwise, reject the message.
(8) If receiver is in the same domain as the sender, the message is forwarded following flow shown in Figure 2.
(9) If receiver is in different domain than the sender, MTA does the following:
  (a) MTA calculates signature of $s_2$ by encrypting it with $SK\_Sender$ — private key of sender's Administrative Management Domain (ADMD) email server: $s_3 = E_{SK\_Sender}(s_2)$.
  (b) MTA appends $s_3$ to the end of message body.

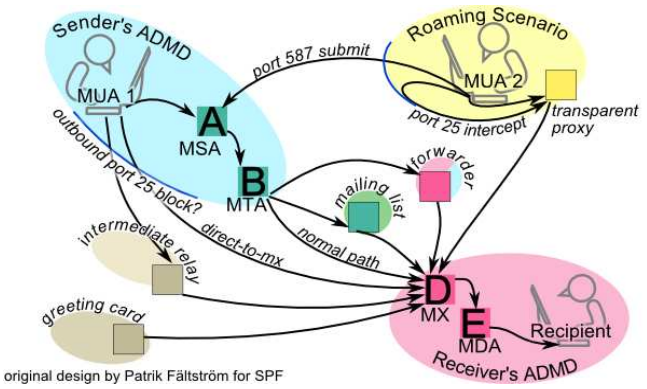



**Figure 2: A schematic representation of the most common ways that an email message can get transferred from its author to its recipient ([2])**

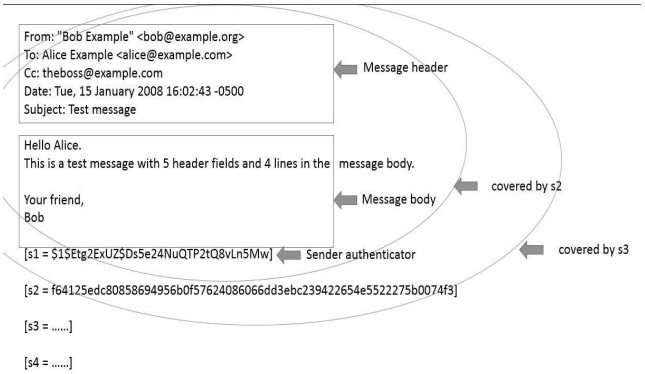



**Figure 3: An email header and body with MAC tag and signature components**

(c) MTA appends $s_4$ — public key certificate of sender's ADMD email server — to the end of message body.

Correspondingly, the process of receiving an email is as follows:

(1) MDA checks if the email is from the same domain. If it is, go to Step 5.
(2) MDA reads signature component $s_4$, and checks with well-know Certificate Authority (CA) to determine if $s_4$ is authentic for sender's ADMD (more is discussed in Section 4). If not, reject the email and stop.
(3) MDA gets sender's ADMD public key $PK\_Sender$ from its certificate $s4$.
(4) MDA reads signature components $s_2$, $s_3$, and verifies if $s_3$ is a valid signature of $s_2$ signed by sender's ADMD specified in $s_4$, by decrypting $s_2$ with $PK\_Sender$ and comparing the result with $s_3$: $s_3 \stackrel{?}{=} D_{PK\_Sender}(s_2)$. If not, reject the email and stop.
(5) MDA reads the message header $mh$, message body $mb$, and signature component $s_1$.
(6) MDA uses $h$ in $s_1$ as key, and uses a predefined MAC algorithm, such as $HMAC$, to calculate a message tag $s_2{}'$ of the message header $mh$ and message body $mb$ and the above $s_1$: $s_2{}' = HMAC_h(mh+mb+s_1)$; and compare it with $s2$ from the email: if $s_2{}' \stackrel{?}{=} s_2$. If not equal, reject the email.

Notice that if the email is sent in the same domain, the MDA does not need to check authenticity of sender's ADMD server, but just checks if the sender uses correct username and password as claimed. Otherwise, MDA needs to verify if the email is really from the claimed domain. This is done by verifying the signature of sender's ADMD server using the public key in its certificate. The public key certificate needs to be issued from a publicly accepted and trusted source.

In summary, the solution is an interlocked chain: we require the email sender provide correct username and password to generate $s1$, use $s1$ as authenticator of the user who is sending the email, and generate $s2$; we use $s2$ to protect integrity of the email and the authenticator, and generates $s3$; we use $s3$ to protect integrity of $s2$; finally, we use $s4$ to verify authenticity and integrity of $s3$, and $s4$ itself is a publicly accepted and trusted public key certificate.

## 4 DISCUSSION

From above description of the solution, we can see that the security of the solution is based on "chain of trust" that is similar to the widely used "certificate chain" but the last segment employs existing technology and infrastructure. The verification of the sender is passed all the way down to the receiver. Since all the connections are interlocked with each other, the receiver and sender can establish end-to-end trust.

We want to bring the following features and limitations of this solutions to the notice of readers:

- *End-to-end authentication and integrity*: The identity of the sender is verified and signed by sender's ADMD email server. The MAC tag and signature cannot be tampered or forged without detection. If User A is sending an email with name User B, MTA can detect the mismatch in sending password and rejects the email. If the sender's domain name is forged by the sender, MTA can also detect and reject it. If an email is intercepted and forged en route, it can be detected by receiver using sender's ADMD email server's signature and public key ceritificate.
- *End-to-end confidentiality*: This solution is designed to ensure authenticity and integrity, by using MAC and signature. If end-to-end confidentiality is required, the sender needs to share a secret key with receiver, or the sender needs to know the public key of the receiver, which needs to be distributed before creating the email and used to encrypt the email body. The email content is transparent to the solution. This

means that an encrypted email body is treated the same way as a plaintext email body.

- *Security of authenticator*: If somebody gets your account password, s/he can send emails in your name — this is the normal case understood by everybody. Password is not secure enough to ensure exclusive authentication. More reliable methods include hardware authentication token, biometric authentication such as fingerprint, iris, face recognition, voice recognition etc. that can provent most of password-based impersonation. The proposed solution is compatible to these choices, as long as the sender's ADMD email server supports them.
- *Public key certificate access*: In the above description, we have the public key certificate of the sender's ADMD email server attached in the email data. This is not efficient and adds traffic overhead, especially for frequent contacts. A better way is to adopt DKIM approach: have the public key certificates (X.509) stored with DNS records of the sender's domain, and retrieve them when needed; a local certificate deposit is also suggested, as most web browsers do.
- *Performance impact*: We are aware that the MAC/ signature generation and verification will lower down performance of the sender's and receiver's servers. But we don't think that's a big concern, as email is not a time-sensitive communication measure, and a little percentage increase of process time does not matter much for normal users, but matters much for senders of batch spamming emails and phishing emails. That's why we did not carry out performance simulation and analysis.

## 5 CONCLUSION

Spam emails and phishing emails are widely existing issues, and most of them are spoofed emails. We consider lack of end-to-end infrastructure the main challenge when fighting against these issues. In the solution presented in this paper, we connect end users to existing security infrastructure, by having the sender's server verify and endorse its users, and passing this verification and endorsement to the receiver. The message tag and signature appended to the email provide authentication and integrity protection to the email. The solution is add-on, which means that it can be implemented on top of existing protocols as an optional component, without replacing email servers and routers in the Internet infrastructure. All work is to be done on sender's and receiver's email client application and sever application.

We believe this is a novel, feasible, and promising solution in current situation and the near future.

## REFERENCES


[1] Dave Crocker. *DKIM Frequently Asked Questions, Version: 16-Oct-2007 10:32*. Available online at www.dkim.org/info/dkim-faq.html.

[2] Patrik Faltstrom. 2017. Most common ways that an email message can get transferred from its author to its recipient. (March 2017). Available online at en.wikipedia.org/wiki/Email_authentication.

[3] Tony Hansen, Dave Crocker, and Phillip Hallam-Baker. 2009. Domain Keys Identified Mail (DKIM) Service Overview. (July 2009). IETF RFC 5585.

[4] S. Kitterman. 2014. Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1. (April 2014). IETF RFC 7208.

[5] J. Klensin. 2008. Simple Mail Transfer Protocol. (October 2008). IETF RFC 5321.

[6] M. Kucherawy and E. Zwicky. 2015. Domain-based Message Authentication, Reporting, and Conformance (DMARC). (March 2015). IETF RFC 7489.

[7] D. Mooloo and T.P. Fowdur. 2013. An SSL-Based Client-Oriented Anti-Spoofing Email Application. In *Africon Conference Proceedings*.

[8] Simon Mundy. *Fraudsters' fingerprints on fake Samsung deal, Financial Times, October 11, 2013. Available online at.* www.ft.com/content/0b972892-3259-11e3-b3a7-00144feab7de.

[9] Jonathan B. Postel. 1982. Simple Mail Transfer Protocol. (August 1982). IETF RFC 821.

[10] A.S. Zadgaonkar, Vikas Chandra Pandey, and Pratap Singh Pradhan. 2013. Authentication against E-Mail Address Spoofing Using Application. *International Journal of Science and Modern Engineering (IJISME)* 1, 6 (May 2013), 13–17.