Chapter – 2
Literature Survey

2.1 Introduction

For over 50 years, email has been a public medium; if we know someone's email address, we can email them [2]. It facilitates seamless interaction between users and a diverse set of desktops, mobile, and web client applications. As proof of its widespread acceptance, an email address is usually required when creating online accounts and making online transactions. As of 2017, there were 6.3 billion email accounts, with 3.7 billion users sending over 269 billion emails every day [2] [12].

Despite its extensive use, email was created without security and is still very insecure today. Email is seldom transmitted via an encrypted connection, with little protection against passive network eavesdropping and active network assaults [2]. There is limited protection against message fabrication [13]. Email archives are frequently huge, stored unencrypted, and vulnerable to hacking. Despite the fact that many email providers filter away spam and viruses, phishing and spear phishing remain challenges [2].

Secure email applications based on S/MIME, such as Microsoft Outlook, have been implemented when there is a strong incentive to protect intellectual property (enterprises) or to meet regulatory requirements (government). However, non-enterprise customers have fewer alternatives [2]. PGP has long been the champion of encrypted email for the public, but it has had low adoption and major usability issues with key management. Several experts are leaving it, including Phil Zimmermann, the founder of PGP; Moxie Marlinspike, who described PGP as a "glorious experiment that has run its course," and Filippo Valsorda, who laments the challenges of holding long-term PGP keys [14]. Academic academics have worked extensively to enhance email PKI for non-enterprise users. Despite this, it has had minimal influence on deployed software after nearly two decades [2].

SMPT is critical to the email system's operation. The easiest approach to describe how SMTP works is to study the sending process, the individual rules and commands that fuel it, and the problems that may occur [15]. Once an SMTP server is set up, email clients can connect to and communicate with it. When the user clicks the "send" button on an email message, the email client establishes an SMTP connection with the server in order to send the message [16]. (The SMTP connection is based on the TCP connection, which stands for Transmission Control Protocol.) From there, the SMTP client utilizes commands to tell the server what to do and transfer data, including the sender's email address, the recipient's email address, and the email's content [15].

A basic structure of an email message is shown in *Figure 2*, which includes the body of the email message, the header consisting of the “From”, “To” & “Subject” fields and the email envelope consisting of the actual “mail from” and “receipt to” fields.

Once an email server receives an email, it checks all the details to see if both email addresses used in the email header and the envelope are from the same email domain. If they are the same as illustrated in *Figure 2,* the email is sent immediately; otherwise, the server utilizes the Domain Name System (DNS) to determine the recipient's domain and sends it to the appropriate server [9].

Figure 2: Basic structure of an email message

Email headers can contain a large number of header fields. Headers typically use the same syntax as the field's name and body, separated by a colon [17]. The header typically includes information about the email's sender and recipient(s), as well as the date and time of transmission. The "Date" and "From" headers are required in all emails. The "To" and "CC" headers are optional; an email can be sent without them, for example, using a BCC (Blind Carbon Copy) email. Email clients conceal the majority of information that would otherwise be visible in raw view. This can include information about a message's delivery, such as the authentication mechanism used, the message's real time of arrival, and which servers the email has passed through [17].

The body of an email is often composed of Multipurpose Internet Mail Extensions (MIME) components [17]. The MIME format frequently contains characters other than the normal ASCII used in plain text communications. MIME also supports media kinds such as photos and audio. The common email protocols include the following:

1. SMTP

The abbreviation SMTP stands for Simple Mail Transfer Protocol, and it is in charge of sending emails. Email clients and servers use this protocol to send emails between computers [18].

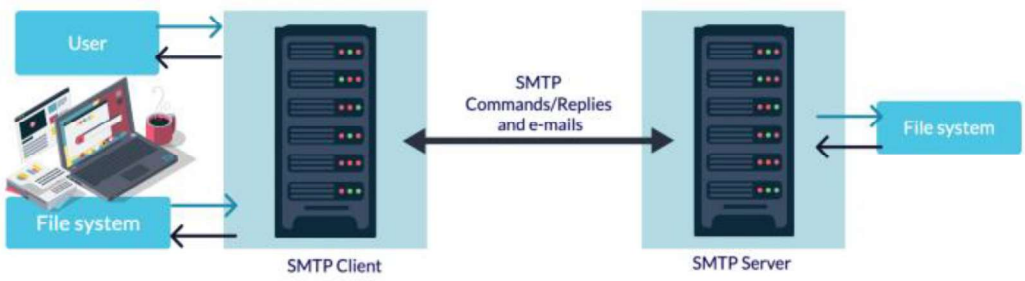

Figure 3: How SMTP Works?

A mail client and SMTP server interact via a specified email port. Both organizations employ SMTP commands and responses to send outbound emails. The Simple Mail Transfer Protocol enables messages to be sent from the same account to many email systems [11]. There are four accessible SMTP ports, each of which supports a distinct sort of email encryption, as described below:

(i) Port 25

Port 25 is the earliest SMTP protocol. In actuality, it goes back to the early days of email communication. When RFC 821 was initially published in 1982, it made Port 25 the default transmission route for internet email. It has remained popular throughout the years and is currently used for numerous transmissions [12].

SMTP port 25 eventually becomes a port for transmitting both legitimate and spam emails. Spammers roamed freely, transmitting massive amounts of spam with malware via this somewhat obsolete port [19][12]. As a result, several hosting companies and ISPs started blocking port 25 for mail input.

Port 25 is currently mostly used for SMTP relaying, the act of delivering messages between email servers. It is not recommended for email submission unless you have your own mail server. When sending emails in this method, you may frequently receive an error message saying that port 25 is closed for email submission. Fortunately, there are other options, including more recent versions that may be freely utilized instead. It is important to remember that SMTP and port 25 were not initially protected [12].

Secure Sockets Layer (SSL) was launched in 1995, becoming the first publicly available technique of email encryption. At the time, adding greater protection to a port did not just mean more work. It was essential to designate a separate port for encrypted communications and one for plain text messages (25). It wasn't only SMTP—modern protocols like FTP, IMAP, and POP use two ports for encrypted and plain text communications [12].

(ii) Port 465

Port 465 was chosen as the new, secure SMTP port for email submission, with port 25 used for relaying. Many platforms quickly migrated there. Not long after, the Internet Assigned Numbers Authority (IANA) and the Internet Engineering Task Force (IETF) decided to reassign this port for new use and propose alternative ports for secure communication [12].

Because of this unusual transition, many services that had lately switched to 465 were left with a deprecated port. Others quickly moved to different ports. Even though port 465 was retired in 1998, we still observe numerous services using this old software. Some publications may even mention it as a recommended port.

SSL has also become obsolete since then, thanks to the rise of TLS (Transport Layer Security), a more advanced approach to security. Many platforms continue to rely on SSL and are required to utilize an SMTP port for SSL, such as 465. Their lack of support for STARTTLS makes it impossible to use other, often superior, ports.

(iii) Port 587

As previously noted, email transmission over port 465 was terminated in 1998. With RFC 2476, the internet authority made port 587 the standard. At the same time, conventional processing was broken into two stages: submission and relay. Relaying was delegated to the good old port 25, and all contributions were now to be routed through port 587 [12].

To this day, the default SMTP port is 587, which should be utilized whenever feasible. In fact, practically all Internet service providers support it. It's also a recommended STARTTLS port, giving emails an extra layer of security over 456. To check if port 587 works on a server, type the following command into the terminal: "telnet example.com 587". The result of "250 STARTTLS" indicates that it is ready [20].

(iv) Port 2525

An ISP or hosting provider may not always support port 587. Other instances, executing "telnet example.com 587" might cause problems. This is where port 2525 can help. It is an alternate port to 587 that supports SMTP with TLS. It has the same features as its elder brother port, but it has never been officially recognized by internet authorities. Its lack of recognition does not prevent it from becoming a viable alternative to 587 [12].

In practice, most ISPs support it. Before a handshake can be exchanged, both parties must establish a connection [13]. TLS provides two techniques to initiate communication:

During a Handshake, an email client utilizing **Opportunistic (Explicit) TLS** notifies the email server that it wants to interact secretly. In other words, it will recommend converting from a plain, unencrypted SMTP connection to a TLS-encrypted one. If the attempt fails, the transmission will start in plain text with no encryption applied.

With **Forced (Implicit) TLS**, an email client will need them to communicate in secret (and via an encrypted connection). An email server will then respond, saying whether or not this configuration is suitable for it. If it is incompatible with the client's TLS version, does not support TLS at all, or the connection fails, the transmission will be terminated, and the email will not be sent further [13].

When using Opportunistic TLS, the SMTP instruction STARTTLS is used to establish an encrypted connection. Despite its name, it can be used to initiate an SSL switch if both sides support the protocol [12].

Almost all email clients that provide TLS encryption to their users include Opportunistic TLS by default. Forced TLS can frequently be enabled at the user's request. The option is between maximum deliverability (Opportunistic TLS) and maximum privacy (Forced TLS). With the first approach, we risk sending some (most likely very few) emails without encryption. The second one may have a modest decline in deliverability [13].

2. POP3

POP3 (Post Office Protocol version 3) allows users to access an email server's inbox. It performs the download and deletion of messages [11]. When a POP3 client connects to the mail server, it retrieves all messages from the mailbox. It then saves the files to the local computer and deletes them from the distant server.

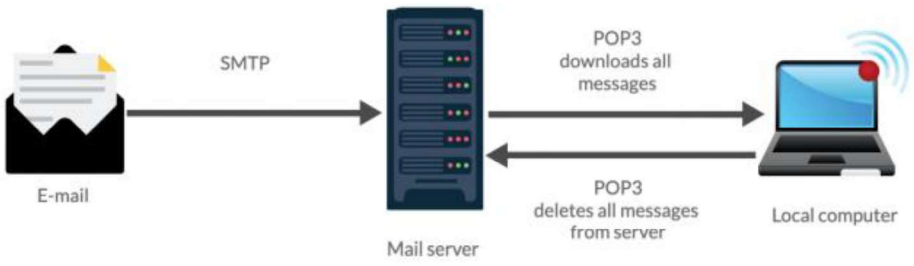

Figure 4: How POP3 works?

This protocol also allows us to access messages locally even in offline mode. Modern POP3 clients allow us to maintain a copy of our messages on the server if we deliberately select that option. Port 110 is the default POP3 port, and it is not encrypted. The encrypted port for POP3 is 995, which uses TLS/SSL. [11].

3. IMAP

The Internet Message Access Protocol (IMAP) allows us to access and manage email messages on the server. This protocol allows us to alter folders, permanently delete messages, and easily search through them. It also allows us to alter or delete email flags and selectively retrieve email attributes. By default, all messages remain on the server unless the user actively deletes them [11].

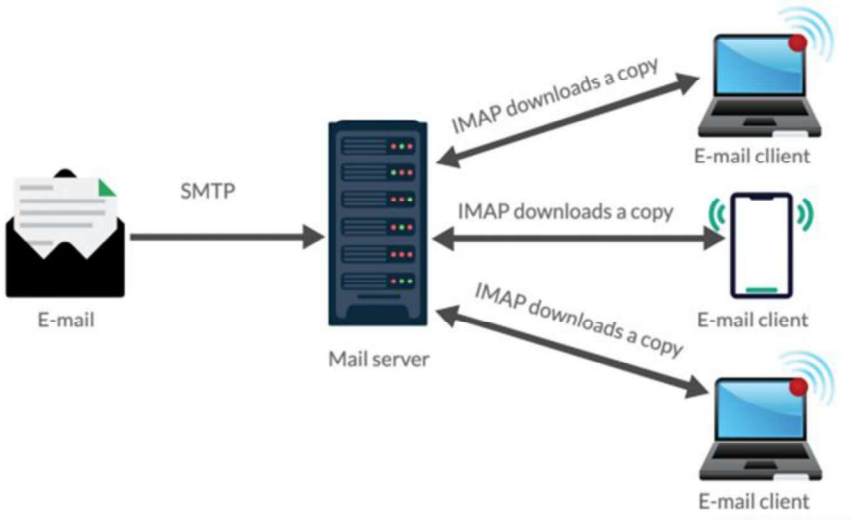

Figure 5: How IMAP works?

IMAP enables multiple users to connect to a single email server. Port 143 is the default port, and it does not support encryption. IMAP's safe port is 993, which is encrypted using TLS/SSL [11].

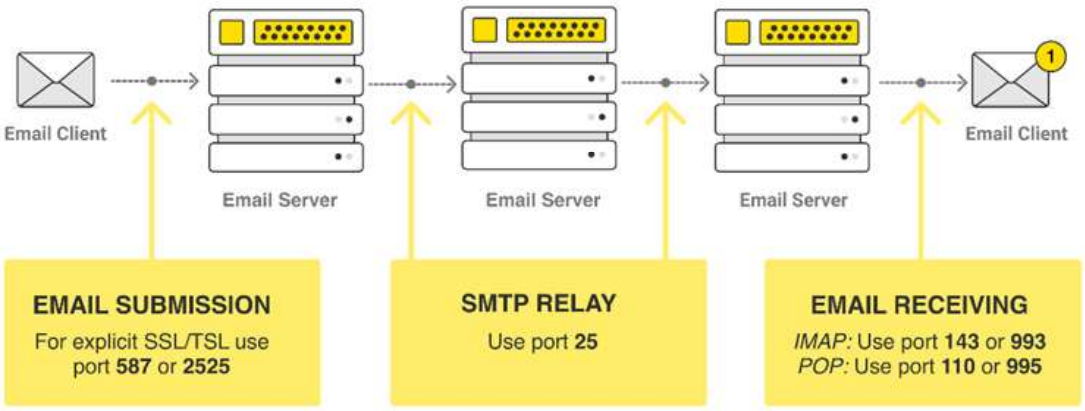

Figure 6: Usage of various ports at different stages of email delivery

If one is establishing their own mail servers to accept mail from external systems (such as Gmail, Yahoo, and so on), they must allow mail to be delivered using the standard port 25. This is the only port that mail servers can use to transfer or relay mail [21].

2.2 Email Security

The poor adoption of encrypted email is usually contrasted with the extensive use of certain messaging applications. WhatsApp and Facebook Messenger have nearly a billion users, while iMessage, Signal, Telegram, Line, and Viber all have millions. The finest of these are forward secrecy, message deniability, and end-to-end encryption. However, despite some calls to abandon encrypted email in favor of Signal, there are compelling reasons to keep using email. Unlike messaging's walled gardens, email is an open system that allows for a variety of uses, including lengthier messages, archiving, searching, and attachments. Email is expected to be the dominant form of communication on the Internet for many years to come, and users of the service may benefit from more privacy. Our study contributes significantly to the literature, which already includes a well-organized body of information concerning secure texting [2].

Service requirements such as universal interoperability, archiving, and searching stymie email security techniques. For example, the user can get public keys in a single administrative domain in an almost transparent way. However, planning for interoperability with hundreds of disparate and uncoordinated administrative domains throughout the world creates significant challenges [2]. The premise that anybody may email anyone else directly needs spam and virus filtering, as well as phishing prevention. Secure messaging alleviates the requirement for these features by restricting who can contact whom. Finally, archive and search make it difficult to provide encryption for webmail or mobile clients since email is frequently kept on a service provider's infrastructure, and unencrypted access is currently required for these activities [2].

Securing email is a tricky issue. Not only is widespread adoption of secure email challenging, but no solution appears to be on the horizon. There are several parties involved, and the multiplicity of use cases for encrypted email makes it impossible to provide a single solution. Our primary objectives are to investigate the history of failures and successes in secure email, including what the term "secure email" even means to various stakeholders; to identify roadblocks and distinguish those that could be removed; to identify tradeoffs among existing systems and clarify how they address different needs; and to use our acquired knowledge to nudge future efforts in fruitful directions [2].

Every aspect of email was first conceived, built, and evolved without consideration for security. As a result, these early designs introduced a plethora of problems, many of which persist today despite decades of work to overcome them. Consider the following example: Email communications are untrustworthy as they were originally written. This means that anyone may send an email to anyone else with a fake sender email address. To understand how this is possible, distinguish between an email message's two components: the envelope and the content. The envelope contains SMTP instructions, which inform mail servers how to transmit the message. These include connection negotiation (HELO or EHLO), the sender's email address (MAIL FROM), one or more recipient addresses (RCPT TO), and maybe other parameters [2]. The original SMTP specifications do not require any validation of the sender's email address.

In addition, many email servers, known as open relays, were originally configured to accept email from any client. Together, these qualities meant that anyone could falsify an email to anyone else, resulting in the birth of unsolicited email or spam [2]. Furthermore, the message body has a distinct format, which includes the standard "From", "To", "CC", and "Subject" headers. Email clients often display the sender's email address in the "From" header rather than in the SMTP envelope. Again, nothing in the original standards prevents the forging of the "From" header. Email's qualities that expose it to spam also make it vulnerable to phishing and malware transmission [2].

In addition to missing legitimacy, the initial mechanisms for sending, receiving, and delivering email between clients and servers lacked integrity and confidentiality safeguards. All messages were delivered in plain text, which may be intercepted and changed by anybody acting as an intermediary. Email is also vulnerable to privacy intrusions since it lacks confidentiality, deniability, traceability, and ephemerality.

Initially, the SMTP protocol did not guarantee communication confidentiality. STARTTLS was proposed as an SMTP addition to encrypt otherwise clear text messages. IMAP and POP3 provide similar functionality. [15]. STARTTLS utilizes TLS to encrypt all SMTP communication, improving authentication between mail servers and avoiding Man-in-the-Middle (MiTM) attacks. Because STARTTLS complies with the ESMTP extension architecture, a separate port is not required; instead, the standard SMTP port is utilized. This architecture greatly enhances interoperability since it uses standard, well-known communication ports and does not necessitate changes to existing settings. The benefits of STARTTLS have accelerated its widespread deployment, which is also strongly supported by major businesses in the field, like Google, Microsoft, Yahoo, and Facebook [15].

To further encourage provider compatibility, SMTP servers are configured to not require STARTTLS to work. Malicious users can use this information to compromise the security of SMTP via downgrade attacks. [15]. An SMTP server provides STARTTLS to clients as one of several supported services. [15]. If the client chooses to utilize this service, the sender and receiver must first perform a TLS handshake to agree on the encryption scheme and standard secret key.

The SMTP session might continue, or cease based on the outcome of the TLS handshake. In the case that authentication and key agreement are successful, all subsequent communication for the current SMTP session will be encrypted with the agreed key [15].

STARTTLS enhances privacy, however it is not without problems. STARTTLS is an opportunistic encryption system, which means it may be built atop either TLS or SSL, with the protocol chosen by negotiation between the two parties. TLS is widely regarded as secure, despite SSL's well-known weaknesses [15].

Furthermore, another problem of STARTTLS is that the choice to use it or not occurs over a plaintext channel. A potential attack would be to intercept the server's list of available services and modify it such that it does not include the STARTTLS option [15]. As a result, the client will wrongly conclude that the server does not support STARTTLS and will proceed with unencrypted SMTP transmission. To address this issue, the parties could make STARTTLS required, which would exclude any server or client that does not support it, resulting in a lack of interoperability between various MTAs. While such a step improves security, it does not address another issue in STARTTLS, which is the use of possibly misleading certificates to authenticate server and client identities [22].

Self-signed certificates are authorized to maintain provider compatibility, although this exposes the system to Man-In-The-Middle attacks. Mail server certificates might be placed in a hierarchical Public Key Infrastructure (PKI) to address these security problems [15]. It usually means that their certificates were issued and signed by a reputable certifying authority, as stated in the proposed standard RFC 6818. Server administrators can deliberately opt to require certificates to be thoroughly inspected (i.e., their authenticity and that of the issuing authority) in order to increase security, but this comes at the price of compatibility with providers that do not employ these certificates [15].

While employing STARTTLS to encrypt the communication channel is a significant improvement over traditional SMTP, it should not be viewed as a panacea. The channel's encryption may be improperly set. It is also important to note that the channel is only encrypted once every hop.

Because email frequently travels via several mail servers before reaching its destination, all traversed servers should employ STARTTLS to ensure that encryption is enforced during the route. Furthermore, the mail server does not encrypt the email. As a result, a hacked mail server might be used to get unauthorized access to users' emails. Users might employ end-to-end encryption technologies such as PGP or S/MIME to protect the secrecy of their email conversations, but this would require effort and understanding. [16]. Furthermore, the difficulty of easy integration of such technologies into major email clients does not help with their widespread adoption [15].

2.3 Email Spoofing

Email spoofing is a contemporary take on the old art of deceit. A falsified email is a sort of impersonation designed to trick recipients into potentially harmful interactions [5]. It is a con that does not require the con artist to engage with the victim in person since it is based on pre-existing relationships. Confidence building is still vital for a successful hoax, but not to the same level as previous approaches. It stands apart from all other sorts of crime, particularly heavy rackets [5].

Email spoofing is a contemporary take on the old art of deceit. A falsified email is a sort of impersonation designed to trick recipients into potentially harmful interactions [5]. It is a con that does not require the con artist to engage with the victim in person since it is based on pre-existing relationships. Confidence building is still vital for a successful hoax, but not to the same level as previous approaches [23]. It stands apart from all other sorts of crime, particularly heavy rackets [5]. Frauds demand a specific degree of education, language skills, and experience. Victims of these schemes do not necessarily engage in fraud on their own. They are frequently honest, not very gullible, and are misled by a well-crafted hoax [5].

Consider the following scenario. According to krebsonsecurity.com, an employee named Judy narrowly missed losing $315,000 in cash from her workplace, a mid-sized manufacturing firm in northeast Ohio [5]. Judy's supervisor had emailed her, asking her to send money to China to pay for some raw materials. The CEO, who was on vacation abroad at the time, had requested far bigger transfers to plants in China and elsewhere, so the request did not look unusual or odd. After Judy had submitted the transfer

instructions to the finance department, she remembered something from the email: The message was far more professional than her boss's typical tone of speech when conversing via email. When she went to investigate the message and learned that an imposter had fooled her, it was too late: the person in charge of initiating wire transfers at her company had already sent it to the bank. Fortunately, the bank hadn't yet processed the transaction, allowing them to retrieve the money. Judy's firm would have suffered a huge loss if the email spoofers had not used such a formal tone, if Judy had been too preoccupied to question the email, or if the bank had executed the transaction more swiftly. Unfortunately, such a favorable set of circumstances does not always exist, and this sort of fraud occurs much too frequently [5].

Email spoofing is a form of social engineering. Defeating security through people rather than technology is the most cost-effective technique and may lead to some of the highest success rates. These attacks take advantage of user apathy or confusion about adequate security practices, tricking them into enabling malicious software applications to enter. Malicious applications are not needed for email spoofing. Unsuspecting loyal personnel may inflict harm [5].

It is crucial to remember that the user is typically vulnerable to a well-executed spoof. Company programs can teach users to be careful of phishing schemes, links, and files, but email spoofing is even more difficult to detect. In a June 2016 post, AOL advised its members that "while there isn't a way to stop whoever is spoofing your account right now, changing your password can help secure your account" [5]. Changing your password may prevent email hacking, but it will not prevent spoofing. The essay goes on to discuss what few email services are doing to avoid server-side spoofing, but a user can only be watchful. Faithful vigilance against email spoofing would involve suspicion of every email, an unachievable goal [24] [5].

Email spoofing is an attack in which an attacker sends an email that looks to be from a trustworthy source other than the intended sender. Email spoofing is a prevalent method used in spam campaigns and phishing attacks because it provides the recipient with the impression that the email is from a real sender. The main goal of email spoofing is to fool recipients into opening or reacting to an email received by a faked sender.

Email spoofing attempts to deceive people into believing that the email is from someone they know or can trust. Once the receiver is satisfied with the email's author, the attacker inquiries about some information that he can use in some way.

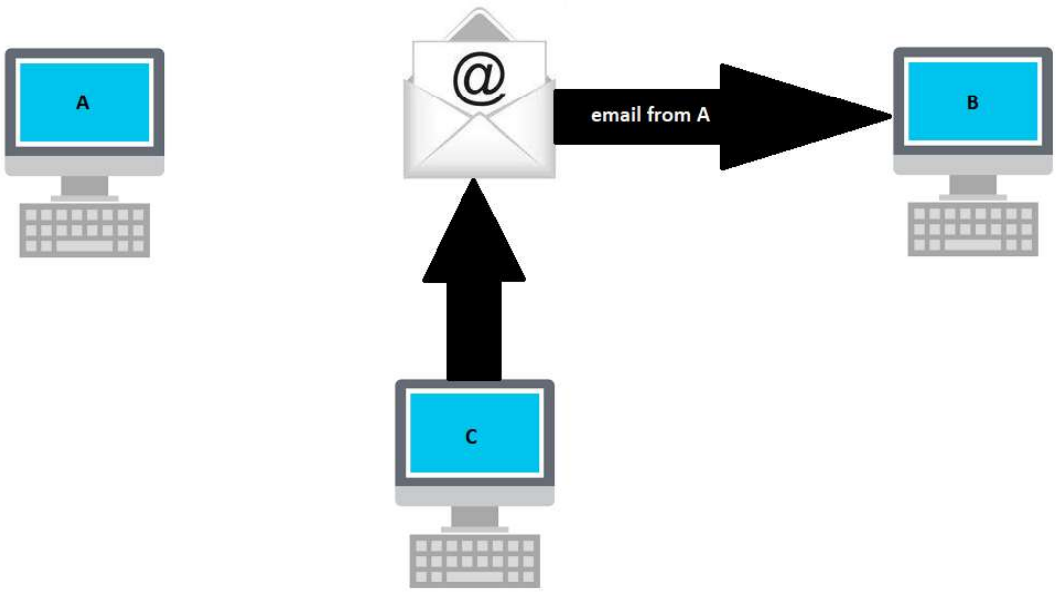

Figure 7: Email Spoofing Process

As shown *Figure 7*, a Spoofed email is sent by person “C” to person “B”, claiming it to have come from person “A”. Here, in this case, person “A” is completely unaware of the email received by person “B”. Now, when person “B” gets this spoofed email, it assumes that this email has come from person “A” as the email of person “A” is mentioned as the sender of the email.

2.4 How Attackers Spoof Email

Email spoofing is done by impersonating email syntax in a number of complex ways. They also differ in terms of the part of the email the attacker will fake [17]. Here are some variants we saw when browsing the web:

1. Spoofing via Display Name

Display name spoofing is an email spoofing technique that involves fabricating solely the email sender's display name. Someone can achieve this by opening a new Gmail or other account with the same name as the person they want to impersonate [25].

If you've ever received an email from Jeff Bezos requesting you to give money, you've seen an example of spoofing using display name [17].

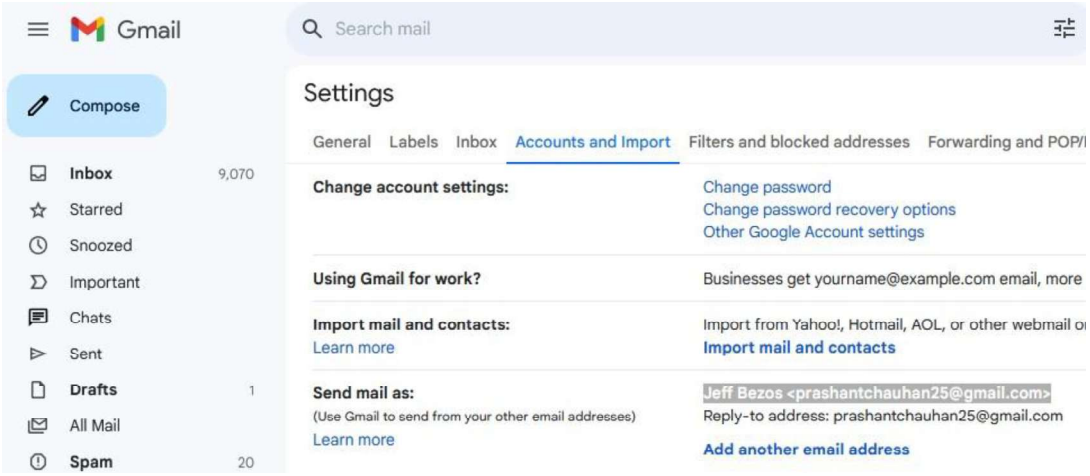

Figure 8: Gmail Settings for changing the display name

As shown in *Figure 9* and *Figure 10,* anyone receiving an email from the email address "prashantchauhan25@gmail.com" will see the name "Jeff Bezos" as the email's sender as per the settings shown in *Figure 8*.

Figure 9: Inbox view having received an email with the fake display name

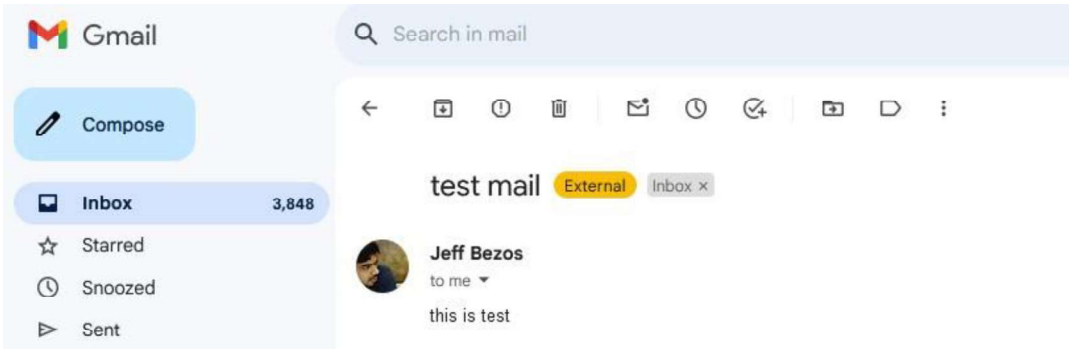

Figure 10: Email details view of an email having a fake display name

This sort of email will also bypass any spoofing security measures. It will not be marked as spam because the email address is legitimate. This takes advantage of user interfaces built for ease of use; most current email client applications do not show information [26]. Because of the extensive usage of smartphone email applications, display name spoofing is fairly common. Often, they just have enough for a display name [17].

2. Spoofing via lookalike domains

Assume a domain is protected, and domain spoofing is not feasible. In that situation, the attacker will most likely create a lookalike domain. In this attack, the fraudster registers and utilizes a domain that is similar to the impersonated domain, such as "@doma1n.com" rather than "@domain.com". An inattentive reader may not notice this alteration. It works because most email receivers don't bother reading the email header and trust what they see as a name in the email.

Original message

Message ID	<20240212055506.77F6B1403E4@mail-server.in>
Created on:	31 January 2023 at 18:12 (Delivered after 32548380 seconds)
From:	Customer Support Executive <prashant@gujaratourism.in>
To:	prashantchauhan25@gmail.com
Subject:	Meeting Reminder 12
SPF:	PASS with IP 117.240.215.158 Learn more

Figure 11: Email message header showing spoofing via lookalike domains

As shown in *Figure 11*, the spoofed email was sent from the domain “gujarat ourism.in”, which looks like the original domain ”gujarattourism. in”. Most recipient users will see this email address as genuine and miss the exact spelling of the original domain name. Also, even if the original domain owner has published its own SPF, DKIM and DMARC records for prevention from spoofing, spoofing is easily possible as the domain name used for sending the email is non-existent and does not have any setup of anti-spoofing protocols like the original one.

The attacker establishes authority by utilizing a similar domain, which also passes spam checks since it has a legitimate mailbox. It may be enough to persuade the victim to provide their password, transfer payments, or share data. In all cases, studying email metadata is the only method to know if the message is genuine. However, this is sometimes tough to do on the fly, especially with smaller smartphone screens [17].

3. Spoofing via Legitimate Domains

Assume the attacker is seeking for greater credibility. In such a situation, he may also include a trustworthy email address in the "From" header, such as "Customer Support Specialist". This implies that both the display name and the email address will include false information [17].

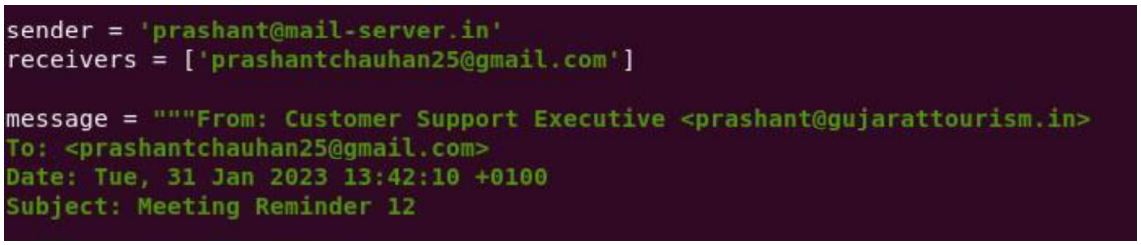

Figure 12: Email message showing spoofing via legitimate domains

As shown in *Figure 12*, the sender's email address and name both are spoofed here to fool the intended recipient of the original sender of the email. This attack does not require to compromise the account or infiltrate the targeted company's internal network. It exclusively exploits hacked Simple Mail Transfer Protocol (SMTP) servers that enable connections without authentication and let users to manually provide the "To" and "From" addresses [17]. This sort of attack allows the attacker to quickly set up a rogue SMTP server and send a fake email.

2.5 How Email Spoofing Works?

A fake email can be made using one of many simple ways. To begin, emails may be faked by visiting an internet spoofing website [5]. “Emkei's Fake Mailer”, “Send Anonymous Email”, “Deadfake” and other similar websites can be found with a Google search. The spoofer does not need to understand any technology [27].

Some servers capture emails from some of these sites and flag them as spam; nonetheless, a serious spoofer will need to raise his game. A more sophisticated email spoof would entail forging the sender's address from the email's header information and making the email appear as close to the spoofed company/person's email as feasible. [5].

Understanding spoofing needs an examination of an email's structure. Email servers create the email header source code, which is subsequently embedded in the email when it is delivered or received. The user sees a "From" box with the sender's email address, a "To" field with his information, a "Subject" field, and the message's text. The "Return-Path" is essentially the bounce address, or the address from which the email was sent [28]. The "Received From" is the transmitting server's IP address; "By" is the server that processed the message; "For" contains the sender addresses; and "Date" is the date the message was transmitted [5]. This may appear many times, depending on the number of email servers involved in the transmission. Some of the key email header fields are as follows:

Authentication-Results: DKIM test results

Received-SPF: indicates SPF test results

To: addressee (visible to the user as the recipient of email)

Subject: email subject as entered by the sender (visible to the user)

From: sender (visible to the user as the sender of the email)

X-Mailer: email client used to send email

Errors-To: bounce back address, should be the sender

Reply-To: recipient of responses to email (visible to the user when they select 'reply')

Message-Id: unique email identifier

Date: self-explanatory

X- etc.: specific email server settings

Spoofing websites allow users to modify the "Return-Path", "From", "Errors-To", "X-Mailer", and "Reply-To" fields. They may also allow encryption, accept the recipient's public key, delay email release until the sender chooses a time, send delivery, and read receipts, add headers, define the SMTP server and port, set priority, and include attachments.

So, we can say that the more detailed a spoofer is, the more convincing the forged email will be [5]. It is critical to understand that faking only three fields is enough to fool any receiving user. Copying emails over telnet requires a bit more expertise, but not much. First, compile a list of MX servers to check for open relays. There are fewer of these currently than in the past, but a fast search yields numerous hits. Other server search utilities include “nslookup” and “dig” [29] [30].

Another alternative is to utilize a web search engine like dnsgoodies.com [5]. Once a list of servers has been built, the user opens a command prompt and enters the following commands:

telnet <possible domain address or IP address – ex mail.server.com> <port - usually 25>
If the connection is successful,
HELO <or **EHLO** depending on the server> <spoofed domain name>
MAIL FROM: <spoofed From email address>
RCPT TO: <To email address>
DATA
SUBJECT: <email subject>
(Enter)
<message>

The most challenging part of the procedure is locating an available relay server. The benefit of telnet is that the Received path displays the available server's domain information rather than faking the website's domain information, making it look more authentic. [31] [32] A critical component of an advanced spoof is ensuring that the spoofer receives any responses to the email rather than the one being impersonated [5]. To do this, many spoofers employ a technique known as domain squatting, which includes registering a domain that is identical to the one used by the person they are impersonating.

They set up an email server and created an email account with a similar domain. This email address is found in the falsified email's Reply-To box. The spoofer believes that the user will not notice the slight variation when they click 'Reply' to respond to the spoofer [5].

To spoof an email, you'll need an SMTP (Simple Mail Transfer Protocol) server and an email application. The attacker changes the "FROM", "REPLY-TO", and "RETURN-PATH" addresses in the message header while the email is being created using this program. Now, regardless of its true origin, everybody who receives this modified email message will assume it came from a forged address [33].

Email spoofing is only possible because SMTP was not initially intended to validate email sender addresses. Although specific methods have been created over time to avoid and detect email spoofing, their acceptance is relatively low [34].

The “Reply-To” field can also be modified or even completely changed by the sender of an email, which can result in spoofing of email phishing attacks. Via the “Reply-To” field, the receiving email server gets to know where to send a reply, and this can be spoofed to be a different email ID instead of the original mailing user’s email address [34] [35].

The receiving email server or even the SMTP protocol cannot identify or take any action to identify the email as fake in circumstances where the reply-to the address and the sender address are different. The receiving server now has full responsibility for comparing the two addresses and deciding whether it believes the email is from a reliable sender. A user risks falling prey to the attacker if they fail to recognize an email from a spoofed sender. Users must analyze its source code before concluding that an email is authentic [36].

The user's IP address who sent the email can be found in the email's source code, which is accessible to the recipient user. From the source code, the user can verify whether the email message has successfully passed the SPF, DKIM, and DMARC tests.

An example of the source code of a forged email can be seen in *Figure 13*, where the authentication test results can be seen under the heading “Authentication-Results”, which, in this case shows SPF as soft fail, DKIM as pass and DMARC as fail. As seen in the figure, the “Return-Path” field has the spoofed email address “prashant@gmail.com”, which is shown to the user in the inbox, while the “Received-From” field has the domain “mail-server. in”, which is the original sending domain.

The main reason that spoofing is easily possible is that the recipient users see the “Return-Path” email address as the email's sender and not the actual sender.

Figure 13: Example of a Spoofed Email

The attacker modifies a few fields in the email message header to spoof emails. A person who wants to spoof an email address can change the address in the "From" field of the email header to any desired email address [37]. Then, as the "Return-Path" field, this fake "From" field is added to the email message header [38] [39].

The email address displayed to the recipient user in their inbox is in the "From" field of the email header [40]. When a person receives a spoof email, they will see the fake "From" email address rather than the real one. Numerous SMTP protocols have been developed to stop and detect email spoofing. The three most efficient ones are Sender Policy Framework (SPF), DomainKeys identified Mail (DKIM) and Domain-based Message Authentication, Routing and Conformance (DMARC) [41] [42]. Although these protocols are still widely used, email spoofing is possible, and attackers can successfully send spoof emails.

2.6 Why Does Email Spoofing Happens?

While email spoofing is frequently used for phishing attacks, there are numerous additional reasons a cybercriminal would try to fake an email account, including [43]:

1. **Anonymity**

   Email spoofing can assist mask the sender's identity, letting them carry out attacks without worrying about the recipient discovering who they are.

2. **Bypassing Spam Filters**

   Most email providers have spam filters, which can assist filter out a large number of spam emails. An attacker may be able to gain access to your mailbox using email spoofing.

3. **Impersonating a Trusted Individual or Organization**

   Email spoofing, like catfishing, can be used to mimic someone you know or a trustworthy institution in the hopes of disclosing personal information that they would not otherwise have access to.

4. **Identity Theft**

   Some faked email communications are intended to fool you into disclosing login credentials or other personally identifiable information, which could lead to identity theft.

5. **Bypassing Block Lists**

   Email spoofing, like evading spam filters, can send a spoofed email to a receiver with whom they would otherwise be unable to communicate.

6. **Spreading Malware**

   A counterfeit email may contain harmful links that download malware, causing damage to your device and jeopardizing your cybersecurity.

7. **Man-in-the-middle (MITM) Attacks**

   Email spoofing is occasionally used to carry out MITM attacks, which include phishing. A common example is when an attacker impersonates your bank by utilizing a bogus sender email address and website URL.

8. **Damaging the sender's reputation**

   Because a faked communication appears to be from someone else, a cybercriminal may use it to destroy the sender's reputation by sending lies or nasty messages.

2.7 How To Identify Spoofed Email?

Email spoofing is extremely hazardous and harmful because it does not require compromising any account by bypassing security protections that most email providers currently apply by default [44]. It takes advantage of the human aspect, primarily because no one double-checks each email's header. Furthermore, it is extremely simple for attackers.

It takes almost no technical know-how to do it on a fundamental level, not to mention that every mail server can be changed to be identical or almost identical to slip by [44].

Furthermore, email faking is easy to detect. Aside from the obvious warning signals, one merely needs to examine the entire email header. It comprises the key components of every email, such as the From, To, Date, and Subject. Metadata will also be utilized to determine how the email was delivered to you and from where it originated. It will most likely contain the verification results the internet service provider used to see if the sender's server has the required authorization to send emails using that domain [44].

How we check this data is extremely reliant on the service we use and will only operate on a desktop. In Gmail, click the three vertical dots next to the reply button and choose "Show Original" from the drop-down list [44].

Some of the strategies used by email servers to alert their recipients of the danger of email spoofing are as follows:

1. **“Via” Warning Que**

   As shown *Figure 14*, a warning cue is displayed in the form of “via mail-server. in”. This indicates to the recipient user that the received email has come via some other email server with a different domain name than the one seen in the received field of the email, i.e., “gujarattourism.in” in this case. Viewing this type of warning cue can be a sign of possible spoofing of email, though it is not at all guaranteed that the email is spam.

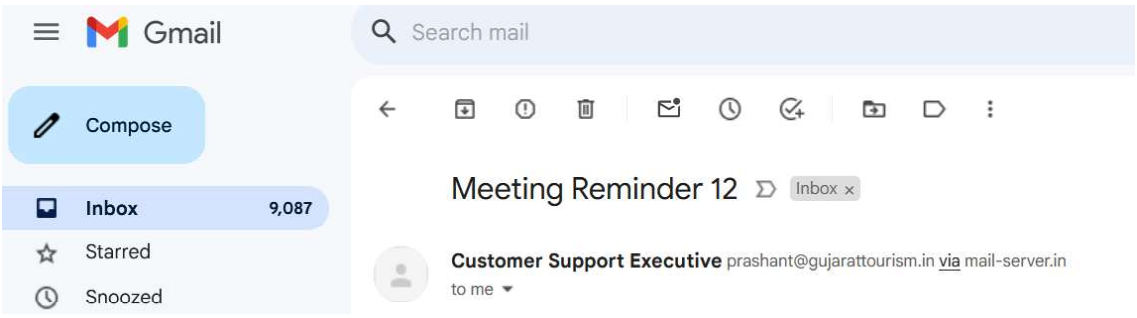

Figure 14: spoofed email showing via in-sender email

2. Warning Message

Figure 15 is an example of a spoofed email that we sent pretending to be a billionaire. In this case, the email filter caught it, labelling it as spam along with a warning “Be careful with the message”, so it didn’t appear in the primary mailbox and instead was delivered to the spam folder.

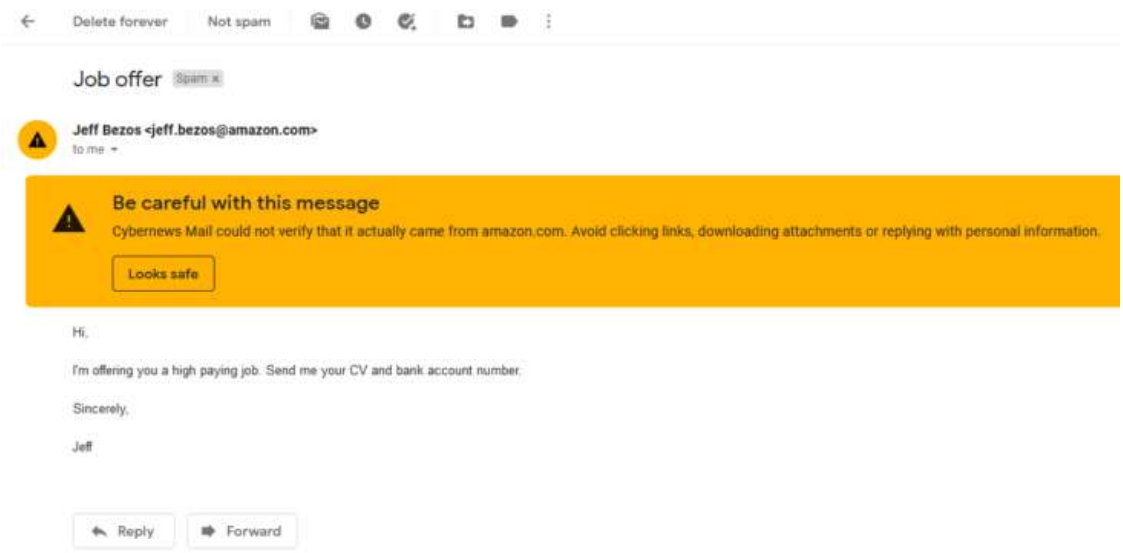

Figure 15: spoofed email showing warning message to user

Apart from these warnings, a recipient user may go to the “Show Original“ menu to find that SPF is indicated as SOFTFAIL, and DMARC is indicated as FAIL, as seen in *Figure 16*. This is enough to identify the email as faked. Some poorly maintained domains do not keep their SPF records up to date and fail validation [44].

Original message

Message ID	<20200819071152.A0133270C7@localhost>
Created on:	19 August 2020 at 10:11 (Delivered after 1 second)
From:	Jeff Bezos <jeff.bezos@amazon.com>
To:	justinas.mazura@cybernews.com
Subject:	Job offer
SPF:	SOFTFAIL with [redacted] Learn more
DMARC:	'FAIL' Learn more

Figure 16: Message header of a spoofed email

If we dig deeper into the code, we can see that the "Received From" and "Received-SPF" domains, as well as the IP addresses, do not match. This is an obvious example of email spoofing. Remember, if the IP addresses do not match and SPF validation fails, this is not a legitimate email. Check if the Return-Path matches the sender's email address [44].

A few of the critical points to detect a spoofed email are as follows [45]:

- The listed sender name does not correspond to the email address.
- The information in the email signature, such as the phone number, is inconsistent with what is known about the sender.
- Check the email header for a RECEIVED line. It should match the email address displayed in the email.
- Check the email header for RECEIVED-SPF. It should state "Pass" If the email states Fail or Softfail, it could have been falsified.
- If the organization uses DKIM and DMARC, the AUTHENTICATION-RESULTS would indicate if the email met the standards of those protocols.
- Warning messages and warning cues in the form of “via” can be used to doubt the possible origin of the email message.
- An email message delivered in spam may be doubted for possible spoofing attack, though not always correct.

2.8 Anti-Spoofing Protocols

To identify and prevent email spoofing, SMTP extension protocols such as SPF, DKIM, and DMARC were developed. The Internet Engineering Task Force (IETF) has published or standardised all three protocols [5].

2.8.1 Sender Policy Framework (SPF)

The Sender Policy Framework (SPF) was first developed in early 2000 and later codified in 2014. SPF allows the owner of an Internet domain to specify which computers can send mail from addresses inside that domain using Domain Name Server (DNS) records. The list of approved transmitting hosts and IP addresses for a domain is published in its DNS records as a specially prepared TXT record.

For instance, the domain "abc.com" may publish its SPF record in the DNS. When the receiving server receives the "MAIL FROM" command claiming to be "xyz@abc.com", it may check and verify if the sender IP address is listed in the "abc.com" SPF record [4].

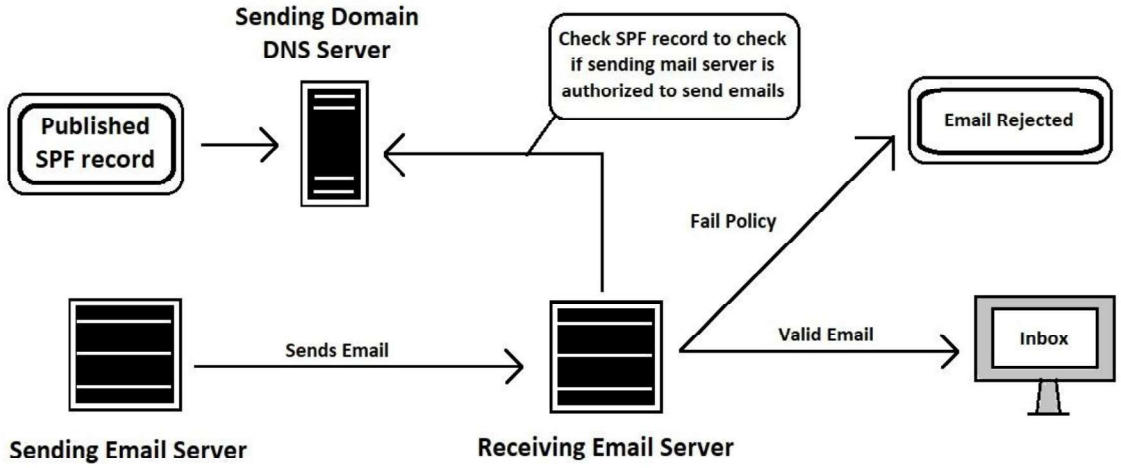

Figure 17: Sender Policy Framework (SPF) Process Flow

Figure 17 shows the Sender Policy Framework (SPF) Process Flow. Receivers that check SPF information in TXT records may reject communications from illegal sources before receiving the message content. Essentially, SPF verifies that the person in the 'Received' field is authorized to send emails from the server from which the emails were sent [5]. As a result, spammers and phishers are less likely to target SPF-protected domains. Because an SPF-protected domain is less desirable as a spoof address, it is less likely to be blocked by spam filters, increasing the likelihood that the domain's real email will be sent.

Advantages of SPF

1. SPF authenticates emails, allowing fraudulent senders to be identified and reported as spam as soon as feasible.
2. Having an SPF boosts the email's reputation.
3. It gives some reassurance that the email is secure and reliable.

Disadvantages of SPF

1. SPF authentication occurs on the chosen Return-Path/Mail-From domain, not the address that most users see. As a consequence, an attacker may send the email from a domain they control using a different sender address. An regular user would not bother to look at the Return-Path/Mail-From, exposing oneself to a phishing assault [20].
2. Email forwarding is a significant issue with SPF. If an email is forwarded by a mail server, the IP address of the forwarding primary server is checked in the original sender's DNS record, which will not be found. As a result, the receiving server may wrongly mark the email as spam.
3. Domain owners typically require that approved third-party suppliers send emails using their domain. Unfortunately, this requires that SPF records be updated whenever an IP address or third-party provider changes. Maintaining these data might be time-consuming [20].
4. Each SPF record allows for ten DNS lookups. If your SPF record goes beyond this limit, receiving servers will fail SPF authentication. However, new technologies, such as “PowerSPF”, allow domain owners to optimize and simplify their SPF record to stay under the restriction [46].
5. Several internal filtering algorithms included into mailbox providers employ the SPF and DKIM protocols to evaluate if an email should be routed to the inbox, spam folder, or refused. However, SPF does not allow domain owners to tell MBPs on how to treat a message if the authentication checks are not validated [46].

SPF's capacity to prevent domain spoofing is limited, so adopting SPF alone provides no protection against email fraud. When used in conjunction with DKIM and DMARC technologies, it can provide effective anti-spoofing security.

2.8.2 DomainKeys Identified Mail (DKIM)

DKIM (DomainKeys Identified Mail) is an email authentication system that detects email spoofing. It was initially drafted in 2004 and standardized in 2011 [47]. It enables the recipient to verify that an email purporting to be from a given domain was actually authorized by the domain's owner.

Its purpose is to prevent counterfeit sender addresses in emails, a tactic commonly employed in phishing and email spam [48] [49].

DomainKeys Identified Mail (DKIM) Process Flow is shown in *Figure 18*. DKIM employs a public-key-based technique to authenticate the email sender and ensure the email's integrity. To maintain integrity and validity, the receiver can verify the DKIM signature to determine whether the signed message has been updated [50]. This tool seeks to ensure that an email remains unchanged while it travels from sender to receiver.

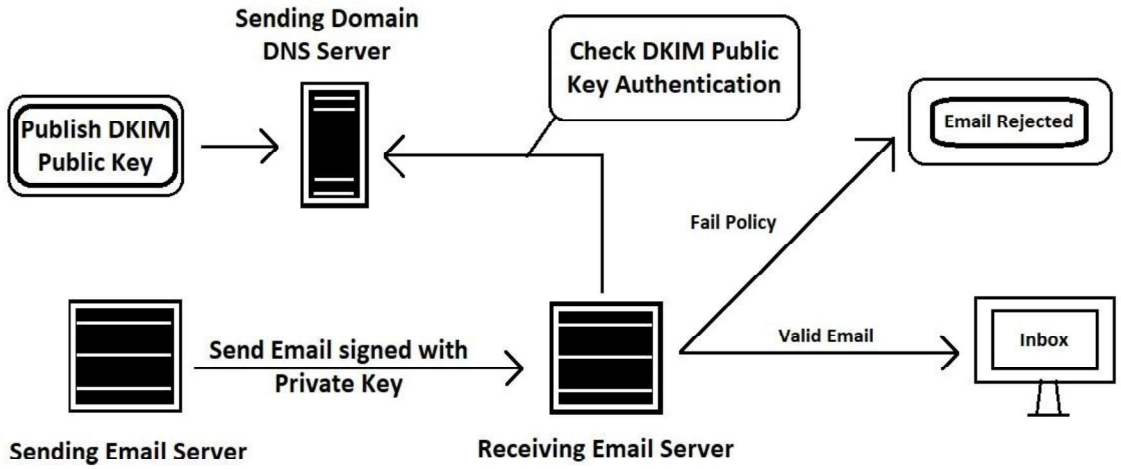

Figure 18: DomainKeys Identified Mail (DKIM) Process Flow

On the sender's end, the server signs the email message with a private key derived from a public key. The public key is kept in the DNS. The receiver-side server searches DNS for the public key and then generates a private key using the same method. It compares the public key, the new private key, and the encrypted private key to determine whether the email was delivered intact from the source. Receivers who successfully verify a signature can use the signer's information in a program to prevent spam, spoofing, phishing, and other unwanted behaviour; but the DKIM standard does not mandate the recipient to perform any specific actions. A spoofer cannot forge a DKIM signature. The server will flag a forged email as failing the DKIM check in the header.

The primary advantage of this strategy for e-mail receivers is that it allows the signature domain to precisely identify a stream of legitimate messages, making domain-based blacklists and whitelists more useful [51].

Advantages of DKIM

1. DKIM is a stronger authentication mechanism than SPF since it employs public-key cryptography rather than IP addresses. [46].
2. SPF is a protocol for adding information to message envelopes. When we forward a message, the forwarding server may remove portions of the message's envelope. However, DKIM works better when forwarding because the digital signature is preserved as part of the email header [46].
3. DKIM is an email tagging system; it does not filter or identify spam on its own. However, it can keep spammers from changing message source addresses.

Disadvantages of DKIM

1. Several internal screening algorithms built into Mailbox Providers employ the SPF and DKIM protocols to evaluate if an email should be delivered to the inbox, spam folder, or refused. However, SPF and DKIM do not allow domain owners to direct Mailbox Providers (MBPs) on how to process a message if the authentication checks cannot be validated. [46].
2. There may be complications if the relay or filtering application modifies the messages [52] [53].
3. A malevolent person can create an email from a respected domain, sign it with their own DKIM, and send it to any mailbox. It can be recovered as a signed email and forwarded to an unlimited number of recipients. This allows an attacker to bypass the DKIM and successfully pass the DKIM test, resulting in the email being delivered to the recipient user's mailbox. [46].

2.8.3 Domain-based Message Authentication, Reporting and Conformance (DMARC)

Domain-based Message Authentication, Reporting, and Conformance (DMARC) is an email validation system that detects and prevents spoofing. It was written in 2011 and released in 2015 [46]. It is intended to combat methods widely used in phishing and email spam, such as emails with forged sender addresses that appear to be from legitimate firms. It prohibits unlawful use of the exact domain name in the "From" field of email message headers [54] [55].

As seen in *Figure 19*, DMARC is built on top of two existing mechanisms: SPF and DKIM. It allows a domain's administrative owner to publish a policy describing which technique (DKIM, SPF, or both) is used when sending email from that domain, as well as how the recipient handles issues [56]. It also contains a system for reporting on actions carried out in compliance with the policies.

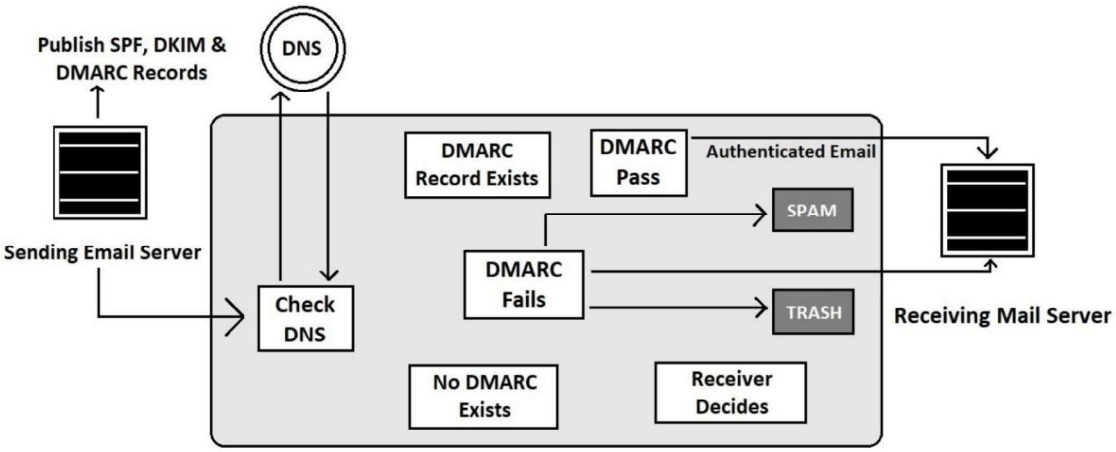

Figure 19: DMARC Process Flow

DMARC coordinates the findings of DKIM and SPF and determines when the "From" header information, which is commonly available to end users, is considered valid. DMARC policies are published in the public Domain Name System (DNS) as text (TXT) resource records (RR), which specify what an email recipient should do with non-aligned mail that arrives [22] [57]. It allows the domain owner to publish a "failing policy" that specifies what action the recipient should take if an incoming email fails DMARC tests [58]. As previously noted, DMARC enables domain owners to choose how MBPs handle unauthenticated communications [59] [60].

Here are a few policy parameters:

- **p=none:** In this policy setting, the email is treated in the same manner as it would be without any DMARC validation.
- **p=quarantine:** With this policy, the email is accepted but directed to a location other than the recipient's inbox, often placed in the spam folder.
- **p=reject:** This policy outright rejects the message, preventing it from being delivered to the recipient's inbox.

To use a DMARC record, we must first enable the SPF and DKIM protocols. Then, following a DMRAC test, we may effectively cover up the following.

- Validation of IP addresses in an SPF record.
- DKIM signature verification.
- Check that the message's From and Return-Path domains are identical.

If the validation fails, the relevant action is done based on the policy defined in the DMARC record, and the report is sent to the supplied email address.

Additionally, DMARC requires identification alignment from SPF or DKIM. DMARC retrieves SPF and DKIM findings and uses them to carry out policies defined by the email server administrator [61] [62]. A DMARC policy allows a sender's domain to declare that their emails are protected by SPF and/or DKIM, and it instructs a recipient what to do if neither of those authentication methods is successful, such as junking or rejecting the message [63] [64]. It removes guesswork from the receiver's processing of these unsuccessful communications, limiting or eliminating the user's exposure to potentially fraudulent or hazardous messages [54].

Advantages of DMARC

- DMARC enables companies and domain owners to receive reports on the email communications they send via the internet [54].
- Having control over your email stream builds trust and adds value to the content you send.
- Make our email clearly traceable across the network of DMARC-enabled receivers.

Disadvantages of DMARC

- Legitimate messages may be blocked or labelled as spam.
- Sometimes the Email servers use liberal of no policy for blocking email which has failed one or both SPF and DKIM, in a fear of not blocking a genuine email which results in delivery of spoofed email in inbox.

2.8.4 Authenticated Received Chain (ARC)

ARC, or Authenticated Received Chain, is a 2016 standard that improves the way DKIM and SPF results are communicated between mail servers during forwarding. When communications are routed through intermediaries like mailing lists or email account forwarding, DKIM and SPF may fail. ARC intends to address this issue. [23].

Email ARC (Authenticated Received Chain) is an email authentication protocol designed to improve the reliability and security of email forwarding. It is an extension of the existing email authentication standards like SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail) and DMARC (Domain-based Message Authentication, Reporting, and Conformance) [65] [66].

When an email is forwarded, the forwarding server adds a new “Received” header to the message, indicating that it has been relayed. However, this can sometimes cause issues with SPF and DKIM authentication, as the original SPF and DKIM signatures may not cover the new Received header. ARC solves this problem by introducing a mechanism for preserving and validating email authentication results across multiple hops or forwarding steps. It allows each intermediary mail server to sign the headers it adds to the email, creating a chain of trust [67].

The recipient’s mail server then verifies these signatures to ensure the message’s integrity and authenticity. The Internet Engineering Task Force (IETF) introduced ARC Protocol in Request for Comments (RFC) 8617 in July 2019 [68]. It’s not perfect, but it adds an additional layer of security that inbox providers need regarding messaging forwarding [69] [70]. ARC preserves the original authentication results from the first hop of an email’s journey and verifies the identity of each intermediate server along the way as shown in the *Figure 20*. The figure shows various ARC headers for an email passing all three anti-spoofing protocols SPF, DKIM and DMARC.

To:	prashant.chauhan-cc@msubaroda.ac.in
Subject:	Keep up to date with research in your field
SPF:	PASS with IP 54.240.53.161 Learn more
DKIM:	'PASS' with domain springer.com Learn more
DMARC:	'PASS' Learn more

Download Original

```
Delivered-To: prashant.chauhan-cc@msubaroda.ac.in
Received: by 2002:a05:6214:5e87:b0:696:59b1:a817 with SMTP id mm7csp1145227qvb;
        Thu, 28 Mar 2024 06:30:58 -0700 (PDT)
X-Google-Smtp-Source: AGHT+IFxE63QVWMr2mXNdEJ4Rw7h1ZlfrivWQcEZ/P8o3tEmKIbE1IKh8D0e0gxHzHM7DLoTFf0P
X-Received: by 2002:a05:6000:cd2:b0:33d:74f2:820e with SMTP id dq18-20020a0560000cd200b0033d74f2820emr1719027wr
        Thu, 28 Mar 2024 06:30:58 -0700 (PDT)
ARC-Seal: i=1; a=rsa-sha256; t=1711632658; cv=none;
        d=google.com; s=arc-20160816;
        b=h1A4uZWAEiEOy4WYMRdy6trrOwBRq+P9Fv3h0GHRMZSe8XcspkJDkHbB2w75C6B38k
         nm4Kvz2KwGRLknDBT3ARyqcnbqAXFqoepSLJEzLQF6FqQnFo7dlDpYjViSiAY2QHj13B
         LodTAMrvABJrXSCxb+pkOFFjjP0CT6/ncH/xveEIQj6faHBK2B2o41JnXrLw0uhT4qXT
         b7fSmMbSVtoqLg0CzXqkiVeRc5kpNNBvCAYIiG55cr3AiC8qv6GqW2zwd+JzrA/rGfLH
         lhawjhgR2MNARVXX6Crf1Zj5MyooTA0gOWTzhmGfZEKQJKqgoQ3QrcIhpcPH6GEQXewa
         VJVw==
ARC-Message-Signature: i=1; a=rsa-sha256; c=relaxed/relaxed; d=google.com; s=arc-20160816;
        h=feedback-id:message-id:mime-version:subject:reply-to:to:from:date
         :dkim-signature:dkim-signature;
        bh=J4463aw1gLBZ6GkBvpZKMtyCfJjdAdwC8XmnoeXZ/S4=;
        fh=b7BQbt/yt/mqei77PWvAGwktZV3bPtS36HMNmdP7yFA=;
        b=EMdaiBIHvBuftKXyMGUJkI0fIJV8h2DVImUq2S2kZt/M4DTrqqHLh20c6PiNjqsE7Q
         DZ6N/TvfaM0loktWLgR5gDktMVLRham5cLtZR/8lS40JXy0EFBxN4QfzbBYLzYETFyTZ
         1iVpKgdj51ZdHmUxbbDQWlrH5cWgTz5wE/OjcOmYvHqeAKaN9/fCa9452/lH1VRsNVqZ
         NwuXSGyKzeaZK0jhB1KYrlQgs3dW82lvDfbG69ldEshI+lOOps0Qku8DASHew7OPQNVR
         5k/t+WjF2aqWdVsTPJM/pyuybxcBrJB/dIQIHgXJFaewtU16ZEwEyUwHDLN/NAy4xz23
         GsBw==;
        dara=google.com
ARC-Authentication-Results: i=1; mx.google.com;
       dkim=pass header.i=@springer.com header.s=j2wowivlc6lbask2l7sjseqtmcfdd5ei header.b=dlKP53x6;
       dkim=pass header.i=@amazonses.com header.s=uku4taia5b5tsbglxyj6zym32efj7xqv header.b=FiXIHmU1;
       spf=pass (google.com: domain of 0102018e8541ac43-5664f361-9b15-40d7-8fac-1e1a9ad9ccca-000000@newsletter.
smtp.mailfrom=0102018e8541ac43-5664f361-9b15-40d7-8fac-1e1a9ad9ccca-000000@newsletter.springer.com;
```

Figure 20: Email Message Details showing ARC Headers

When an email is routed through a trusted intermediate server, the server digitally signs the message and includes the ARC signature in the email header. Each time a message travels from A to B, trustworthy intermediary servers attach their signature, forming a chain of ARC signatures [24].

By validating the "chain of custody" or authenticated received chain, the recipient's email server may examine the initial authentication findings. It can also validate that any modifications to the email in transit were signed by a reliable intermediary [24].

ARC adds three extra email headers as explained in *Table 1* to messages to create a chain of trust back to the original message.

ARC Header	What it contains
ARC-Authentication-Results	Copy of the email authentication results: SPF, DKIM, and DMARC.
ARC-Message-Signature	A digital signature similar to a DKIM signature, encompassing the entire message and headers (excluding the ARC-Seal header).
ARC-Seal	A DKIM-like signature that includes the ARC headers generated by each intermediate server.

Table 1: ARC Headers and their purposes

If an intermediary email server modifies a message, it digitally signs the modification to ensure that it is valid [24]. If the recipient's mail server notices that a message has failed DMARC, it can verify the ARC result by:

1. Validating the chain of ARC-Seal headers
2. Validating the latest ARC-Message-Signature (based on the sequence number)

If everything is correct, the message passes ARC. If the recipient server trusts all intermediate servers in the ARC chain, it may accept the message despite failing DMARC. ARC enables the recipient's server to accept legitimate communications that might otherwise be rejected or designated spam, hence enhancing email deliverability and security [71].

2.9 Gaps in Existing System

Despite these attempts, sending spoofing emails is still shockingly simple today [5]. In 2017, global SPF deployment was reported to be 53.8%, DKIM at 38.8%, and DMARC at 46.8%. Most email administrators are aware of the technical flaws of SPF, DKIM, and DMARC, which is one of the reasons for these protocols' low adoption rate.

The prevailing impression is that these protocols are "helpful," but "cannot fully solve the spoofing problem". Some of the key limitations found in the existing system are as follows:

1. **Problem of Identifier Alignment in SPF and DKIM**

   Both SPF and DKIM suffer from "identifier alignment" issues. It means that the sender email address that the user sees may differ from the address used to accomplish authentication. For SPF, authentication focuses on the "Return-Path" and checks to see if the sender's IP address is listed in the "Return-Path" domain's SPF records [72] [73]. To pass authentication, an attacker can set the "Return-Path" domain to his own domain and update his SPF record. This is feasible because the "From" field, rather than the "Return-Path" field, determines what the recipient sees on the email interface.

   DKIM has a similar issue in that the domain used to sign the email with the DKIM key may differ from the domain specified in the "Return-Path". DMARC helps to resolve the problem by mandating the alignment of IDs [5].

2. **Mail Forwarding is a Problem for SPF**

   Mail forwarding is the process of automatically forwarding emails from one email service to another. A common occurrence is that corporate employees configure their workplace email service to forward all emails to Outlook or Gmail. During mail forwarding, the email metadata (such as "Return-Path") remains unaltered. SPF fails during mail forwarding because the forwarder's IP does not match the original sender's SPF record [5].

3. **Mailing List is a Major Problem for both SPF and DKIM**

   When you send a message to a mailing list, it will "broadcast" it to all of its subscribers. This is a similar approach to mail forwarding. During this process, the IP address of the mailing lists is changed to that of the sender, which differs from the original sender's address. This will cause SPF failure.

Mailing lists will also cause issues with DKIM because most mailing lists modify email content before sending it to subscribers. The most typical alteration is to include a "footer" with the mailing list name and an unsubscribe link. Tempering the email content will result in DKIM failure. DMARC helps to fix some of the difficulties, but not the mailing list issue. For mailing lists, DMARC+SPF is certain to fail: if the "Return-Path" is changed, DMARC will fail due to identifier misalignment; if the "Return-Path" remains unchanged, SPF will fail owing to an IP mismatch. DMARC+DKIM will fail if the mailing list needs to modify the email content.

4. **Lack of Critical Mass**

Because of the technique's shortcomings, the widespread opinion is that SFP, DKIM, and DMARC are "helpful" but "cannot completely solve the spoofing problem," which is why these protocols have a poor adoption rate. If everyone doesn’t use these protocols the overall effect is diminished.

5. **No Penalty for not Publishing SPF, DKIM, and DMARC**

There is no penalty for domains that do not publish an SPF/DKIM/DMARC record. Their emails are usually not discriminated against unless other malicious signals are identified. As a result, not everyone is encouraged or motivated to actively use these standards.

6. **Benefits not Significantly Overweight Costs**

The protocol adopter receives no direct benefit by posting their SPF, DKIM, or DMARC records in the DNS. Instead, these DNS records are primarily used to help other email services validate incoming emails and protect their clients (users). Domains that publish the DNS records gain from a better reputation, which is a somewhat imprecise advantage. For non-email domains (e.g., office.com), posting the SPF/DMARC record prevents attackers from spoofing the domain and helps the domain maintain a positive reputation. Domain administrators publish SPF/DMARC records to be a good Internet "citizen" and assist other email providers in detecting spoofing emails. However, these benefits are regarded indirect, making them relatively weaker.

7. Lack of Control on the DNS or Mail Servers

Some services do not have control over their DNS records. Publishing an SPF/DKIM/DMARC record will require additional effort to communicate with their DNS providers. Furthermore, many businesses and organizations use cloud-based email services rather than maintaining their own mail servers. The firm must rely on the cloud email service to implement anti-spoofing mechanisms.

8. Risks of Breaking the Existing System

Email providers need to go through careful testing to make sure that the protocol does not block legitimate incoming emails, and their own emails are not blocked by others. This is the reason why most protocol adopters configure a relaxed SPF/DMARC policy. Even if sender domain specified a strict protocol, the receiver may not enforce it anyway.

9. Perception of Difficulty

There is a widespread impression that implementing anti-spoofing measures is difficult. Regardless of the real amount of difficulty, the perceived difficulty discourages email administrators from even trying.

10. Lack of External 3rd Party Verification of SPF Records

Anyone can sign the SPF records for a domain. Verisign, for example, examines websites to ensure their security when they sign up for SSL. If your website isn't good, you won't get "Verified by Verisign". However, there is no corresponding "Signed by SPF" authority to ensure that whoever signs up for it genuinely qualifies to receive it [6]. A fraudster might use this weakness to get around the SPF element of email security by simply registering a domain on rented servers (perhaps in another country) and adding their own SPF record to it. Thus, when the destination server sends a DNS query, the domain checks out, and the email goes unmarked through the SPF system [6].

11. Typo-Squatted Domain

A spoofer can set up an email server behind a typo-squatted domain and create his own DKIM public/private key system. This will evade the DKIM protections, hence it is best utilized as part of a tiered defence scheme.

2.10 Conclusion and Summary

In this chapter we started with the email system briefly covering the process of delivery of an email from source user to destination user. We also discussed email security, specifically the authentication of email and then we discussed email spoofing, what is email spoofing, why email spoofing happens and how does email spoofing actually happen. Then we covered the various anti-spoofing protocols namely Sender Policy Framework (SPF), Domainkeys Identified Mail (DKIM), Domain-based Message Authentication, Reporting and Conformance (DMARC) along with their working process, advantages and disadvantages in detail. At the end we noted various gaps in existing system of email delivery which lays the foundation to our research work.