Лабораторна робота № 2

Сервіс електронної пошти

1. Короткі теоретичні відомості.

Електронна пошта (email або e-mail) - це метод обміну повідомленнями (поштою) між людьми, які використовують електронні пристрої. У 1960-х роках електронна пошта використовувалася обмежено, користувачі могли надсилати її лише користувачам одного комп’ютера. Деякі системи також підтримували форму миттєвих повідомлень, коли відправник і одержувач повинні знаходитися в режимі онлайн одночасно. Рей Томлінсон вважається винахідником електронної пошти; у 1971 році він розробив першу систему, здатну надсилати пошту між користувачами на різних вузлах по ARPANET, використовуючи знак @ для зв’язку імені користувача з сервером призначення. До середини 1970-х років цю форму визнали електронною поштою.

Електронна пошта працює через комп’ютерні мережі, насамперед через Інтернет. Сучасні системи електронної пошти базуються на моделі зберігання та пересилання. Поштові сервери приймають, пересилають, доставляють та зберігають повідомлення. Ні користувачі, ні їхні комп’ютери не зобов’язані перебувати в мережі одночасно; їм потрібно підключитися, як правило до поштового сервера або через вебінтерфейс, щоб надсилати або отримувати повідомлення або завантажувати їх.

Спочатку як текстовий комунікаційний носій ASCII, електронна пошта була розширена за допомогою багатоцільових розширень Інтернет-пошти (MIME) для перенесення тексту в інших наборах символів та вкладеннях мультимедійного вмісту. Міжнародна електронна пошта з інтернаціоналізованими адресами електронної пошти, що використовують UTF-8, стандартизована, але не набула широкого поширення.

1.1. Формат повідомлення.

RFC 822 визначає повідомлення, які мають дві частини: заголовок та тіло. Обидві частини представлені в тексті ASCII. Первинно тіло повідомлення було простим текстом. Це все ще так, хоча RFC 822 був доповнений MIME, щоб дозволити тілу повідомлення нести різноманітні типи даних. Ці дані все ще представлені у вигляді тексту ASCII, але оскільки вони можуть бути кодованою версією, скажімо, зображення JPEG, вони не обов’язково читаються користувачами.

Заголовок повідомлення являє собою ряд рядків, які закінчуються символами <CRLF>. (<CRLF> позначає повернення каретки плюс подачу рядка, які є парою керуючих символів ASCII, які часто використовуються для позначення кінця рядка тексту.) Заголовок відокремлений від тіла повідомлення порожнім рядком. Кожен рядок заголовка містить тип і значення, розділені двокрапкою. Багато з цих рядків заголовків знайомі користувачам, оскільки їх просять заповнити, коли вони складають повідомлення електронної пошти; наприклад, заголовок ідентифікує одержувача повідомлення, заголовок говорить щось про мету повідомлення. Інші заголовки заповнюються базовою системою доставки пошти. Приклади включають (якщо повідомлення передане) ім'я користувача, що надіслав повідомлення та кожен поштовий сервер, який обробляв це повідомлення. Звичайно, є багато інших рядків заголовків описаних в RFC 822.

RFC 822 було розширено в 1993 році (і з тих пір неодноразово оновлювалось), щоб дозволити повідомленням електронної пошти нести багато різних типів даних: аудіо, відео, зображення, документи PDF тощо. MIME складається з трьох основних частин. Перший фрагмент являє собою набір рядків заголовків, які доповнюють вихідний набір, визначений RFC

822. Ці рядки заголовка різними способами описують дані, що переносяться в тілі повідомлення. Вони включають: версію MIME, що використовується, зручний для читання опис того, що є в повідомленні, типи даних, що містяться в повідомленні, і як дані в тілі повідомлення закодовані.

Друга частина - це визначення набору типів вмісту (та підтипів). Наприклад, MIME визначає кілька різних типів зображень, включаючи `image/gif` та `image/jpeg`, кожен із очевидним значенням. В якості іншого прикладу, `text/plane` посилається на простий текст, який ви можете знайти у звичайному повідомленні, тоді як `text/richtext` позначає повідомлення, що містить текст із «розміткою» (текст із використанням спеціальних шрифтів, курсиву тощо). Як третій приклад, MIME визначає тип програми, де підтипи відповідають результатам роботи різних прикладних програм (наприклад, `application/postscript` та `application/msword`).

MIME також визначає тип `multipart`, який говорить про те, як структуровано повідомлення, що містить більше одного типу даних. Це як мова програмування, яка визначає як базові типи (наприклад, цілі числа та числа з плаваючою комою), так і складені типи (наприклад, структури та масиви). `mixed` один із можливих `multipart` підтипів, який говорить, що повідомлення містить набір незалежних фрагментів даних у визначеному порядку. Потім кожен фрагмент має власний рядок заголовка, який описує тип цього фрагмента.

Третя частина - це спосіб кодування різних типів даних, щоб вони могли бути відправлені в електронному повідомленні ASCII. Проблема полягає в тому, що для деяких типів даних (наприклад, зображення JPEG) будь-який 8-бітовий байт зображення може містити одне з 256 різних значень. Лише підмножина цих значень є дійсними символами ASCII. Важливо, щоб повідомлення електронної пошти містили лише ASCII, оскільки вони можуть проходити через низку проміжних систем (шлюзи, як описано нижче), які передбачають, що вся електронна пошта є ASCII, і це може пошкодити повідомлення, якщо воно містить символи, що не є ASCII. Для вирішення цієї проблеми MIME використовує пряме кодування двійкових даних у набір символів ASCII. Кодування називається base64. Ідея полягає в тому, щоб відобразити кожні три байти вихідних двійкових даних у чотири символи ASCII. Це робиться шляхом групування двійкових даних у 24-бітові одиниці та розбиття кожної такої одиниці на чотири 6-бітові частини. Кожен 6-розрядний фрагмент відображається на одному з 64 дійсних символів ASCII; наприклад, 0 відображається на A, 1 на B і так далі. Якщо подивитись на повідомлення, закодоване за допомогою схеми кодування base64, то можна побачити лише 52 великі та малі літери, 10 цифр від 0 до 9 та спеціальні символи + та /. Це перші 64 значення в наборі символів ASCII.

Окрім того, щоб зробити читання пошти якомога безболіснішим для тих, хто все ще наполягає на використанні лише текстових зчитувачів пошти, повідомлення MIME, яке складається лише зі звичайного тексту, може кодуватися за допомогою 7-бітового ASCII. Також є читабельне кодування для даних ASCII.

Поєднавши все це, повідомлення, що містить звичайний текст, зображення JPEG та файл PostScript, виглядатиме приблизно так:

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="-------417CA6E2DE4ABCAFBC5"
From: Alice Smith <alice@cisco.com>
To: bob@comsys.kpi.ua
Subject: test message
Date: Mon, 06 Sep 2021 19:45:19 +0200
```

```
---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Bob,

Here are the jpeg image and draft report I promised.

--Alice

---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
... unreadable encoding of a jpeg figure
---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit
... readable encoding of a PostScript document
```

У цьому прикладі рядок у заголовку повідомлення говорить, що це повідомлення містить різні фрагменти, кожен позначений рядком символів, який не відображається в самих даних. Потім кожен фрагмент має власні рядки Content-Type та Content-Transfer-Encoding.

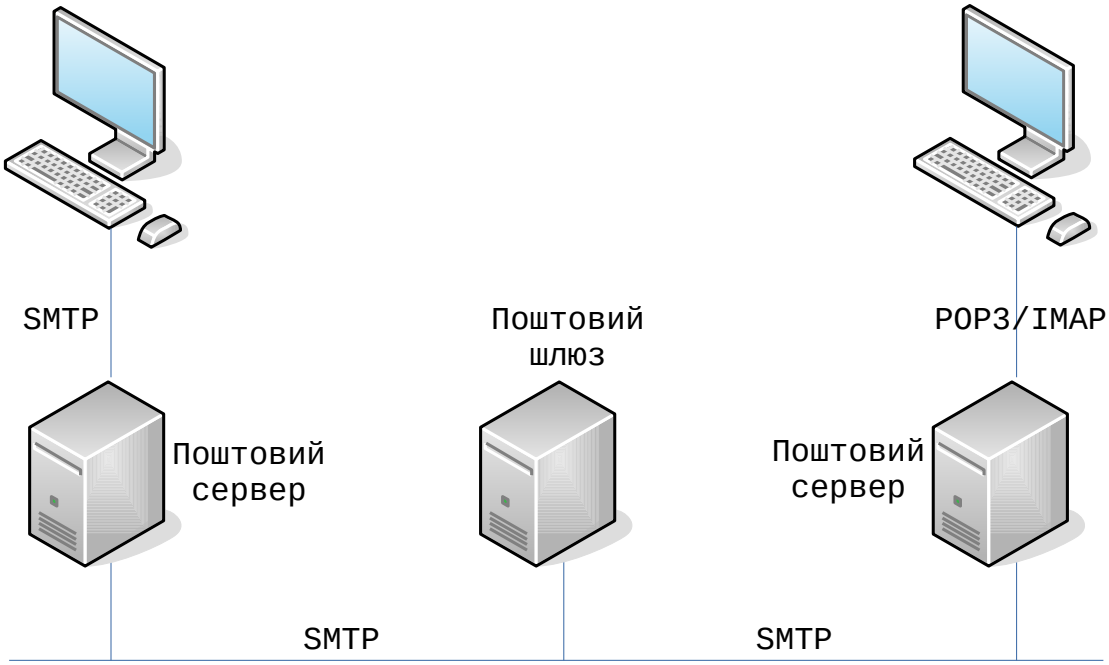
1.2. Передача повідомлень.

Протягом багатьох років більшість електронних листів переміщувались з вузла на вузол, використовуючи лише SMTP. Незважаючи на те, що SMTP продовжує відігравати центральну роль, зараз це лише один із декількох протоколів електронної пошти. Інтернет-протокол доступу до повідомлень (IMAP) та Протокол поштового відділення (POP) є ще двома важливими протоколами отримання поштових повідомлень.

Щоб зрозуміти роль SMTP у правильному контексті, необхідно визначити ключових гравців. По-перше, користувачі взаємодіють із програмою редагування пошти, коли вони складають, зберігають, шукають та читають свої електронні листи. Існує велика кількість редакторів пошти, як і багато веб-браузерів. На початку становлення мережі Інтернет користувачі, як правило, входили на вузол, на якому знаходилась їх поштова скринька, а програма зчитування пошти, яку вони викликали, була локальною програмою, яка завантажувала повідомлення з файлової системи. Сьогодні, звичайно, користувачі віддалено отримують доступ до своєї поштової скриньки зі свого ноутбука чи смартфона; вони не входять на вузол, що зберігає їхню пошту (поштовий сервер). Другий протокол передачі пошти, такий як POP або IMAP, використовується для віддаленого завантаження електронної пошти з поштового сервера на пристрій користувача.

По-друге, на кожному вузлі, що містить поштову скриньку, працює поштовий демон (або процес). Цей демон називається агентом передачі повідомлень (MTA), який відіграє роль поштового відділення. Користувачі (або їх редактори пошти) передають демону повідомлення, які вони хочуть надіслати іншим користувачам, демон використовує запущений SMTP через TCP для передачі повідомлення демону, що працює на іншій машині, і демон розміщує вхідні повідомлення в поштовій скриньці користувача (де редактор пошти цього користувача може їх пізніше знайти). Оскільки SMTP - це протокол, який може реалізувати кожен, теоретично може існувати безліч різних реалізацій поштового демона. На практиці широко використовується

лише декілька популярних реалізацій, причому найпоширенішими є стара програма sendmail від Berkeley Unix, postfix та exim.



Малюнок 1. Послідовність пересилання повідомлень електронної пошти.

Хоча, безумовно, можливо, що MTA на машині відправника встановлює SMTP/TCP-з'єднання з MTA на поштовому сервері одержувача, у багатьох випадках пошта передається через один або кілька поштових шлюзів на своєму шляху від вузла відправника до вузла одержувача. Як і на кінцевих вузлах, ці шлюзи також використовують демон агента передачі повідомлень. Не випадково ці проміжні вузли називаються шлюзами, оскільки їх робота полягає у зберіганні та пересиланні повідомлень електронної пошти, подібно до того, як “шлюз IP” (який називається маршрутизатором) зберігає та пересилає пакети IP. Єдина різниця полягає в тому, що поштовий шлюз зазвичай буферизує повідомлення на диску і готовий спробувати повторно передати їх на наступний вузол протягом декількох днів, тоді як IP-маршрутизатор буферизує пакети в пам'яті і готовий повторити спробу їх передачі лише за долю секунди. Малюнок 1 ілюструє шлях із одним проміжним вузлом на шляху від відправника до одержувача.

Для чого потрібні поштові шлюзи? Чому вузол відправника не може надіслати повідомлення вузлу одержувача? Одна з причин полягає в тому, що одержувач не хоче вказувати конкретний вузол, на якому він читає електронну пошту, що надходить на його адресу. Інша причина - це масштаб: у великих організаціях часто трапляється, що на різних вузлах зберігаються поштові скриньки організації. Наприклад, пошта, що надсилається на bob@comsys.kpi.ua, спочатку може надсилатися на поштовий шлюз mail.comsys.kpi.ua, а потім пересилається - за допомогою другого з'єднання до конкретного вузла, на якій bob має поштову скриньку. Шлюз переадресації підтримує базу даних, яка відображає користувачів у вузли, на яких знаходяться їх поштові скриньки. Відправник не повинен знати цього конкретного імені. (Список заголовків рядків у повідомленні допоможе простежити шлюзи пошти, по яких пройшло певне повідомлення.) Ще однією причиною, особливо на ранніх етапах розвитку електронної пошти, є те, що вузол, на якому розміщується поштова скринька будь-якого користувача, не завжди може бути доступний і в цьому випадку поштовий шлюз зберігає повідомлення до моменту його доставки.

Незалежно від кількості поштових шлюзів на шляху, незалежне з'єднання SMTP використовується між кожним вузлом для переміщення повідомлення ближче до одержувача. Кожен сеанс SMTP включає діалог між двома демонами пошти, причому один виконує роль клієнта, а інший - сервера. Під час одного сеансу між двома вузлами може передаватися кілька повідомлень. Оскільки RFC 822 визначає повідомлення, що використовують ASCII як базове представлення, SMTP також базується на ASCII. Це означає, що користувач може видавати себе клієнтською програмою SMTP.

SMTP найкраще зрозуміти на простому прикладі. Далі наводиться обмін між відправляючим доменом comsys.kpi.ua та приймаючим доменом cisco.com. У цьому випадку користувач bob з comsys.kpi.ua намагається надіслати пошту користувачам alice та tom в cisco.com. Додано додаткові порожні рядки, щоб зробити діалог більш читабельним.

```
HELO pc.comsys.kpi.ua
250 mail.comsys.kpi.ua Hello pc.comsys.kpi.ua [10.18.51.101]

MAIL FROM:<bob@cs.comsys.kpi.ua >
250 OK

RCPT TO:<alice@cisco.com>
250 OK

RCPT TO:<tom@cisco.com>
550 No such user here

DATA
354 Start mail input; end with <CRLF>.<CRLF>
Hello Alice and Tom!
<CRLF>.<CRLF>
250 OK

QUIT
221 Closing connection
```

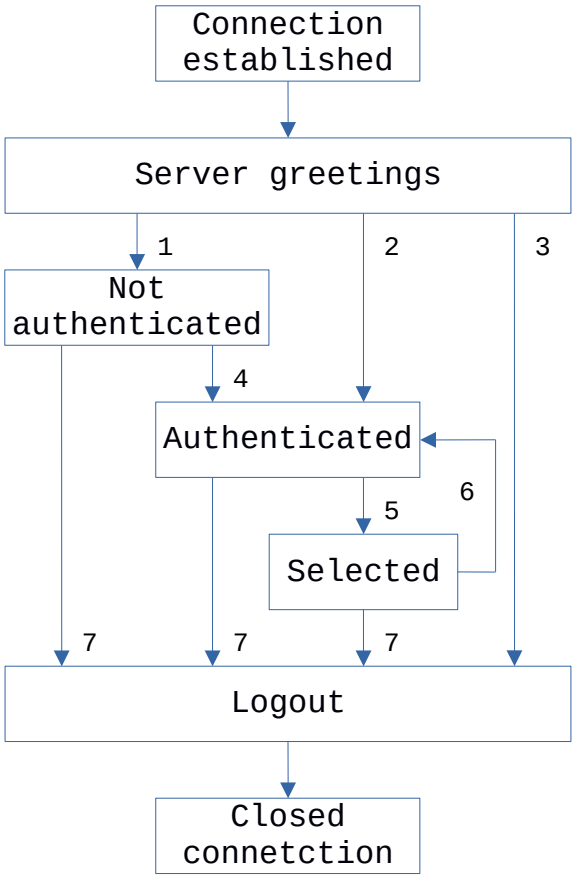
Як видно, SMTP передбачає послідовність обмінів між клієнтом та сервером. У кожному обміні клієнт публікує команду (наприклад, QUIT), а сервер відповідає кодом (наприклад, 250, 550, 354, 221). Сервер також повертає зручне для читання пояснення коду (наприклад, "`No such user here`"). У цьому конкретному прикладі клієнт спочатку ідентифікує себе на сервері за допомогою команди HELO. В якості аргументу він надає своє доменне ім’я. Сервер перевіряє, що це ім'я відповідає IP-адресі, що використовується TCP-з'єднанням (сервер повідомляє цю IP-адресу клієнту). Потім клієнт запитує у сервера, чи готовий він прийняти пошту для двох різних користувачів; сервер відповідає, кажучи "так" одному, а "ні" іншому. Потім клієнт надсилає повідомлення, яке закінчується рядком із єдиною крапкою ("."). Нарешті, клієнт розриває з'єднання.

Звичайно, існує безліч інших команд і кодів повернення. Наприклад, сервер може відповісти на команду клієнта RCPT кодом 251, який вказує на те, що користувач не має поштової скриньки на цьому вузлі, але що сервер обіцяє переслати повідомлення на інший поштовий демон. Іншими словами, вузол функціонує як поштовий шлюз. В якості іншого прикладу, клієнт може здійснити операцію VRFY для перевірки електронної адреси користувача, але фактично не надсилаючи повідомлення користувачеві.

1.3. Редактор пошти.

Останній крок - це те, що користувач фактично отримує свої повідомлення з поштової скриньки, читає їх, відповідає на них і, можливо, зберігає копію для подальшого використання. Усі ці дії користувач виконує, взаємодіючи з програмою редагування пошти. Як зазначалося раніше, спочатку цей зчитувач був просто програмою, що працює на тій самій машині, що і поштова скринька користувача, і в цьому випадку він міг просто читати та писати файл, який реалізує поштову скриньку. Це було поширеним випадком в епоху до ноутбуків. Сьогодні найчастіше користувач отримує доступ до своєї поштової скриньки з віддаленої машини, використовуючи ще один протокол, такий як POP або IMAP.

IMAP багато в чому схожий на SMTP. Це протокол клієнт/сервер, що працює через TCP, де клієнт (який працює на комп'ютері користувача) видає команди у вигляді <CRLF> -термінованих текстових рядків ASCII та поштового сервера (працює на вузлі, яка підтримує поштову скриньку користувача). Обмін починається з того, що клієнт автентифікує себе та ідентифікує поштову скриньку, до якої він хоче отримати доступ. Це може бути представлено простою діаграмою переходу стану, зображеною на малюнку 2. На цій діаграмі LOGIN і LOGOUT - це приклади команд, які може видавати клієнт, тоді як OK - одна з можливих відповідей сервера. Інші загальні команди включають і EXPUNGE, з очевидними значеннями. Додаткові відповіді сервера включають NO (клієнт не має дозволу виконувати цю операцію) та BAD (команда неправильно сформована).



Малюнок 2. Діаграма переходу стану IMAP.

Коли користувач відправляє команду `FETCH`, сервер повертає повідомлення у форматі MIME, а програму редагування пошти декодує. Окрім самого повідомлення, IMAP також

визначає набір атрибутів повідомлення, якими обмінюються як частина інших команд, незалежно від передачі самого повідомлення. Атрибути повідомлення включають таку інформацію, як розмір повідомлення та, що ще цікавіше, різні прапори, пов’язані з повідомленням (наприклад, `Seen`, `Answered`, `Deleted` та `Recent`). Ці прапорці використовуються для синхронізації клієнта та сервера; тобто, коли користувач видаляє повідомлення у програмі зчитування пошти, клієнт повинен повідомити про цей факт поштовий сервер. Пізніше, якщо користувач вирішить видалити всі видалені повідомлення, клієнт відправляє команду `EXPUNGE` серверу, який знає, що насправді видаляє всі раніше помічені на видалення повідомлення з поштової скриньки.

Нарешті, коли користувач відповідає на повідомлення або надсилає нове повідомлення, програма редагування пошти не пересилає повідомлення від клієнта на поштовий сервер за допомогою IMAP, а замість цього використовує SMTP. Це означає, що поштовий сервер користувача є фактично першим поштовим шлюзом, пройденим по шляху від комп'ютера відправника до поштової скриньки одержувача.

2. Завдання на роботу.

2.1. Створити центр сертифікації (CA) issuer і subject якого відповідають варіанту завдання. Сертифікати для поштових серверів мають підписуватись сертифікатом та ключем створеного CA. Сертифікат створеного CA необхідно додати в список довірених на всіх вузлах, які використовуються в роботі.

2.2. Встановити та налаштувати 2 поштових сервери (mail/message transfer agent - MTA) для пересилки повідомлень електронної пошти за допомогою протоколу SMTP, які відповідають наступним вимогам:

- кожний поштовий сервер є кінцевим отримувачем повідомлень для домену, ім'я якого відповідає варіанту завдання;
- доменне ім'я поштового серверу має відповідати формату: mail.<ім'я домену>;
- в кожному домені створені поштові скриньки з іменами, які відповідають варіанту завдання;
- обов'язкова автентифікація користувачів при відправленні повідомлень на зовнішні MTA;
- підтримка TLS для передачі повідомлень;
- вихідні повідомлення містять підпис DKIM;
- перевірка вхідних повідомлень на наявність та коректність підпису DKIM, повідомлення з некоректним підписом відкидаються;
- перевірка домену відправника повідомлення та обробка повідомлення у відповідності з SPF, яка відповідає варіанту завдання;
- перевірка повідомлень на наявність шкідливого програмного забезпечення;
- контекстний аналіз повідомлень для виконання спам-фільтрації.

2.3. Для кожного сервера додати функціонал доставлення повідомлень засобами протоколів POP3 та IMAP з підтримкою TLS.

2.4. Виконати аналіз протокольного обміну між вузлами під час відправлення, пересилки та доставлення поштових повідомлень. Проаналізувати заголовки листів.

2.5. Рекомендується використовувати наступне програмне забезпечення:

- MTA: exim або postfix;
- доставлення повідомлень засобами POP3 та IMAP: dovecot;
- антивірусна фільтрація: ClamAV;
- спам-фільтрація: SpamAssassin.

2.6. Кожний поштовий сервер рекомендується встановлювати та налаштовувати на окремій віртуальній машині. Для перевірки роботи модулів поштового сервера та аналізу протокольного обміну рекомендується використовувати утиліти: telnet, openssl, swaks.

2.7. Додати необхідні записи типів: MX та TXT для забезпечення функціонування поштових серверів згідно з варіантом завдання.

Варіант	Issuer і Subject CA	Домен 1	Домен 2	Поштові скриньки	Політика SPF
1	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=letter CA	letter.net	zone01.com	alpha@letter.net beta@letter.net gamma@letter.net delta@zone01.com omega@zone01.com	v=spf1 mx -all
2	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=planet CA	planet.edu	zone02.org	mercury@planet.edu venus@planet.edu earth@planet.edu saturn@zone02.org jupiter@zone02.org	v=spf1 mx ~all
3	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=cat CA	cat.com	zone03.net	tiger@cat.com lion@cat.com lynx@cat.com leopard@zone03.net jaguar@zone03.net	v=spf1 a mx -all
4	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=flower CA	flower.org	zone04.edu	rose@flower.org gerbera@flower.org tulip@flower.org aster@zone04.edu peony@zone04.edu	v=spf1 ptr mx -all
5	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=linux CA	linux.net	zone05.com	ubuntu@linux.net debian@linux.net centos@linux.net gentoo@zone05.com fedora@zone05.com	v=spf1 a mx ~all
6	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=color CA	color.edu	zone06.net	red@color.edu green@color.edu blue@color.edu black@zone06.net white@zone06.net	v=spf1 ptr mx ~all
7	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=metal CA	metal.com	zone07.org	gold@metal.com silver@metal.com iron@metal.com copper@zone07.org zinc@zone07.org	v=spf1 mx -all
8	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=capital CA	capital.org	zone08.edu	london@capital.org tokyo@capital.org paris@capital.org rome@zone08.edu berlin@zone08.edu	v=spf1 mx ~all

Варіант	Issuer і Subject CA	Домен 1	Домен 2	Поштові скриньки	Політика SPF
9	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=currency CA	currency.net	zone09.com	dollar@currency.net dinar@currency.net lira@currency.net peso@zone09.com real@zone09.com	v=spf1 a mx -all
10	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=river CA	river.edu	zone10.net	nile@river.edu amazon@river.edu congo@river.edu amur@zone10.net mekong@zone10.net	v=spf1 ptr mx -all
11	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=fruit CA	fruit.com	zone11.org	apple@fruit.com orange@fruit.com grape@fruit.com banana@zone11.org lemon@zone11.org	v=spf1 a mx ~all
12	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=digit CA	digit.org	zone12.edu	one@digit.org two@digit.org three@digit.org four@zone12.edu five@zone12.edu	v=spf1 ptr mx ~all
13	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=month CA	month.net	zone13.com	march@month.net april@month.net may@month.net june@zone13.com july@zone13.com	v=spf1 mx -all
14	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=name CA	name.edu	zone14.org	maria@name.edu tomas@name.edu tereza@name.edu stefan@zone14.org sara@zone14.org	v=spf1 mx ~all
15	C=UA ST=Ukraine L=Kyiv O=KPI OU=OT CN=country CA	country.com	zone15.net	france@country.com china@country.com spain@country.com italy@zone15.net germany@zone15.net	v=spf1 a mx -all

3. Контрольні питання
3.1. Архітектура електронної пошти.
3.2. Призначення та принципи роботи протоколів SMTP, POP3 та IMAP.
3.3. Формат поштового повідомлення.
3.4. Маршрутизації повідомлень електронної пошти.
3.5. Методи автентифікації поштових повідомлень DKIM та SPF.
3.6. Технологія DMARC.

4. Література.

https://web.archive.org/web/20210412192955/https://book.systemsapproach.org/applications/traditional.html

https://en.wikipedia.org/wiki/Email

RFC5321 https://tools.ietf.org/html/rfc5321

RFC1939 https://tools.ietf.org/html/rfc1939

RFC3501 https://tools.ietf.org/html/rfc3501

RFC6376 https://tools.ietf.org/html/rfc6376

RFC7208 https://tools.ietf.org/html/rfc7208