České učení technické v Praze
Fakulta strojní
Ústav přístrojové a řídící techniky

SMTP Proxy server pro zabezpečené odesílání zpráv

Diplomová práce

Vít Bartes

Magisterský program: Automatizační a přístrojová technika
Magisterský obor: Automatizace a průmyslová informatika
Vedoucí práce: Ing. Cyril Oswald, Ph.D.

Praha, červen 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bartes** Jméno: **Vít** Osobní číslo: **476071**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Automatizační a přístrojová technika**

Specializace: **Automatizace a průmyslová informatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

SMTP Proxy server pro zabezpečené odesílání zpráv

Název diplomové práce anglicky:

SMTP Proxy server for secure messaging

Pokyny pro vypracování:

- Proveďte rešerši aktuálního stavu techniky bezpečnosti elektronických zpráv v Internetu
- Proveďte rešerši možností technické realizace v oblasti emailových proxy server
- Navrhněte SW architekturu pro zabezpečené odesílání zpráv na základě provedené rešerše
- Proveďte základní implementaci navržené SW architektury

Seznam doporučené literatury:

[1] J. C. Klensin, "Simple Mail Transfer Protocol," Internet Engineering Task Force, Request for Comments RFC 5321, říj. 2008, Num Pages: 95. doi: 10 . 17487 / RFC5321. URL: https://datatracker.ietf.org/doc/rfc5321

[2] W. Stallings, Cryptography and Network Security - Principles and Practice, 7th Edition. Pearson Education, India,2018. ISBN 9789353942564

[3] H. Finney, L. Donnerhacke, J. Callas, R. L. Thayer a D. Shaw, "OpenPGP Message Format," Internet Engineering Task Force, Request for Comments RFC 4880, lis. 2007, Num Pages: 90. doi: 10 . 17487 / RFC4880. URL: https : / / datatracker . ietf.org/doc/rfc4880.

[4] H. Orman, Encrypted email: the history and technology of message privacy, eng, ř. SpringerBriefs in computer science. Cham, Heidelberg, New York, Dordrecht, London: Springer, 2015, isbn: 978-3-319-21343-9.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Cyril Oswald, Ph.D. U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **29.04.2022** Termín odevzdání diplomové práce: **09.06.2022**

Platnost zadání diplomové práce: _____

Ing. Cyril Oswald, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

doc. Ing. Miroslav Španiel, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studenta

Vedoucí práce:
Ing. Cyril Oswald, Ph.D.
Ústav přístrojové a řídící techniky
Fakulta strojní
České vysoké učení technické v Praze
Technická 4
160 00 Praha 6
Česká republika

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze 2022

.....
Vít Bartes

Abstrakt

Abstrakt

Tato diplomová práce se zaměřuje na vývoj SMTP proxy serveru pro bezpečnou emailovou komunikaci. Je v ní provedena rešerše aktuálního stavu technologií elektronické pošty. Práce se také zaměřuje na bezpečnost elektronické pošty z pohledu dnešních technologií a způsoby zajištění soukromé komunikace. V práci je navržena architektura proxy serveru, který šifruje odchozí poštu pro službu GDPRCloud. Zároveň je popsána i jeho realizace v programovacím jazyce Java včetně způsobu distribuce symetrického klíče.

Klíčová slova: SMTP, Email, GDPRCloud, TLS, OpenPGP, šifrování, autentizace, SMTP proxy

Abstract This thesis focuses on the development of an SMTP proxy server for secure email communication. A survey of the current state of the art of email technologies is conducted. The thesis also focuses on the security of email from the perspective of today's technologies and ways to secure private communication. The thesis proposes a proxy server architecture that encrypts outgoing mail for GDPRCloud service. At the same time, its implementation in Java programming language is described, including the method of symmetric key distribution.

Keywords: SMTP, Email, GDPRCloud, TLS, OpenPGP, encryption, authentication, SMTP proxy

Poděkování

Rád bych touto cestou vyjádřil poděkování Doc. Ing. Josefu Kokešovi CSc. a Ing. Cyrilu Oswaldovi Ph.D. za jejich cenné rady, doporučení a trpělivost při vedení mé diplomové práce. Rád bych poděkoval také své rodině, přítelkyni a všem přátelům, kteří mě při vytváření této práce podpořili, a bez jejichž pomoci by nebylo možné práci dokončit.

Seznam obrázků

Seznam zkratek

AES Advanced Encryption Strandard — specifikace pro šifrování dat. 26, 48

API Application Programming Interface — spojení mezi počítačovými programy. 48, 56, 57

ASCII American Standard Code for Information Interchange — tabulka znaků. 10, 11

DNS Domain Name System — doménový systém. 3, 12, 13

GDPR General Data Protection Regulation — nařízení o ochraně údajů a soukromí. 43

GSM Global System for Mobile communications — mobilní komunikační standard. 57

HMAC Hash-based Message Authentication Code. 19

IETF Internet Engineering Task Force — normalizační organizace. 27, 35

IMAP Internet Message Access Protocol — protokol příchozí pošty. 5, 8, 9, 22

IP Internet Protocol — komunikační protokol. 4, 11, 13, 19, 27

JDK Java Development Kit — distribuce technologie Java. 57

JRE Java Runtime Enviroment — softwarová vrstva, která běží nad operačním systémem počítače. 47

MAC Message Authentication Code — kontrolní kód zprávy. 27, 29, 30

MD5 Message Digest 5 — hashovací funkce. 18

MIME Multipurpose Internet Mail Extensions -víceúčelová rozšíření internetové pošty. 10, 11

MX Mail Exchanger record — záznam o emailovém serveru spojeného z danou doménou. 13

PGP Pretty Good Privacy — šifrovací standard. 26, 35

PII Personally Identifiable Information — osobní data. 26

POP3 Post Office Protocol 3 — protokol příchozí pošty. 5, 8, 9, 19

RSA Rivest–Shamir–Adleman — šifrovací systém s veřejným klíčem. 26, 36

SASL Simple Authentication and Security Layer — autentizační framework. 17

SID Security Identifier — autorizační číslo. 57

SMS Short Message Service — telefonní služba pro zasílání zpráv. 57

SMTP SimpleMail Transfer Protocol — protokol odchozí pošty. 1, 5, 8–12

SSL Secure Sockets Layer — kryptografický protokol. 20, 27

TCP Transmission Control Protocol — komunikační protokol. 9, 11, 12, 19, 22, 27

TLS Transport Layer Security — kryptografický protokol. 2, 14, 20, 26, 27

Obsah

Kapitola 1

Úvod

Elektronická pošta neboli email je systém, který uživatelům umožňuje odesílat a přijímat zprávy prostřednictvím počítačové sítě. Jedná se o jeden z prvních elektronických komunikačních nástrojů dostupný široké veřejnosti.

Vznik komunikačního standardu SMTP (Simple Mail Transfer Protocol — protokol pro přenos elektronické pošty) se datuje do 70. let minulého století. Jedná se tedy o protokol starší než internet sám. Standard SMTP nebyl v průběhu let příliš upravovaný a vylepšovaný. Nevýhodou takto starého protokolu je především jeho nedokonalost v oblasti bezpečnosti. Při původním návrhu emailového standardu probíhala komunikace mezi emailovými servery v prostém textu (nešifrovaně). To představovalo obrovské bezpečnostní riziko. V nynější době ovšem bezpečnost v oblasti počítačové komunikace hraje významnou roli.

Aby mohl být email nadále využíván, byly v průběhu let navrženy různé mechanismy pro zabezpečení.V současné době se zabezpečení emailové komunikace dělí do dvou větví. Bezpečnost při přenosu a bezpečnost mezi koncovými uživateli (tzv. end-to-end).

Většina veřejně známých poskytovatelů internetové pošty dnes nabízí zajištění bezpečnosti při přenosu. To znamená, že komunikace mezi klientem a serverem, popř. mezi serverem a serverem, je šifrovaná. Nedostatek tohoto způsobu tkví v tom, že zpráva na jednotlivých serverech zůstává nezašifrovaná. To představuje potenciální riziko úniku informací. Navíc poskytovatel nedokáže zaručit, že všechny servery, přes které zpráva jde, podporují šifrování při přenosu. Proto se může stát, že zpráva bude zaslána přes internet nezašifrovaná.

Zabezpečení komunikace mezi koncovými uživateli eliminuje tento nedostatek. I přes tento fakt většina poskytovatelů nativně nenabízí tento způsob zabezpečení. V případě, že poskytovatel tuto metodu nabízí, bývá její zprovoznění pro běžného uživatele obtížné. Navíc uživatelé, kteří spolu chtějí komunikovat šifrovaně si musí sami zajistit distribuci svého veřejného klíče.

Na základě výše zmíněných nedostatků se zrodila myšlenka vytvoření architektury, která nativně poskytuje end-to-end šifrování s minimální znalostí uživatele. K tomu má dopomoci emailový proxy server, který udělá veškerou práci, z hlediska bezpečnosti, za uživatele.

1.1 Cíle práce

Cílem této diplomové práce je:

- Seznámení se s architekturou emailové komunikace se zaměřením jak na komunikační protokoly, tak i na jednotlivé softwarové části nutné pro zasílání a příjem elektronické pošty.
- Provézt rešerši v oblasti bezpečnosti elektronické pošty. Z hlediska bezpečnosti přenosu se seznámit s dnes již široce používaným standardem TLS (Transport Layer Security). Zejména se zaměřit na architekturu protokolu a způsob šifrovaní. Z hlediska end-to-end šifrování na možnosti zavedení tohoto typu zabezpečení v elektronické poště a definice standardů.
- Zaměřit se na definici proxy serverů, typy SMTP proxy serverů a na možnosti technické realizace proxy serverů v elektronické poště.
- Ze získaných informací o emailové architektuře a bezpečnosti navrhnout architekturu emailové proxy. Vytvořit objektový model. Dále do navrženého proxy serveru implementovat způsob zajištění end-to-end bezpečnosti. A nakonec celý tento návrh realizovat do softwarového řešení. Nutná podmínka tohoto softwaru je jeho spustitelnost na operačním systému Linux.

Kapitola 2

Emailoví agenti

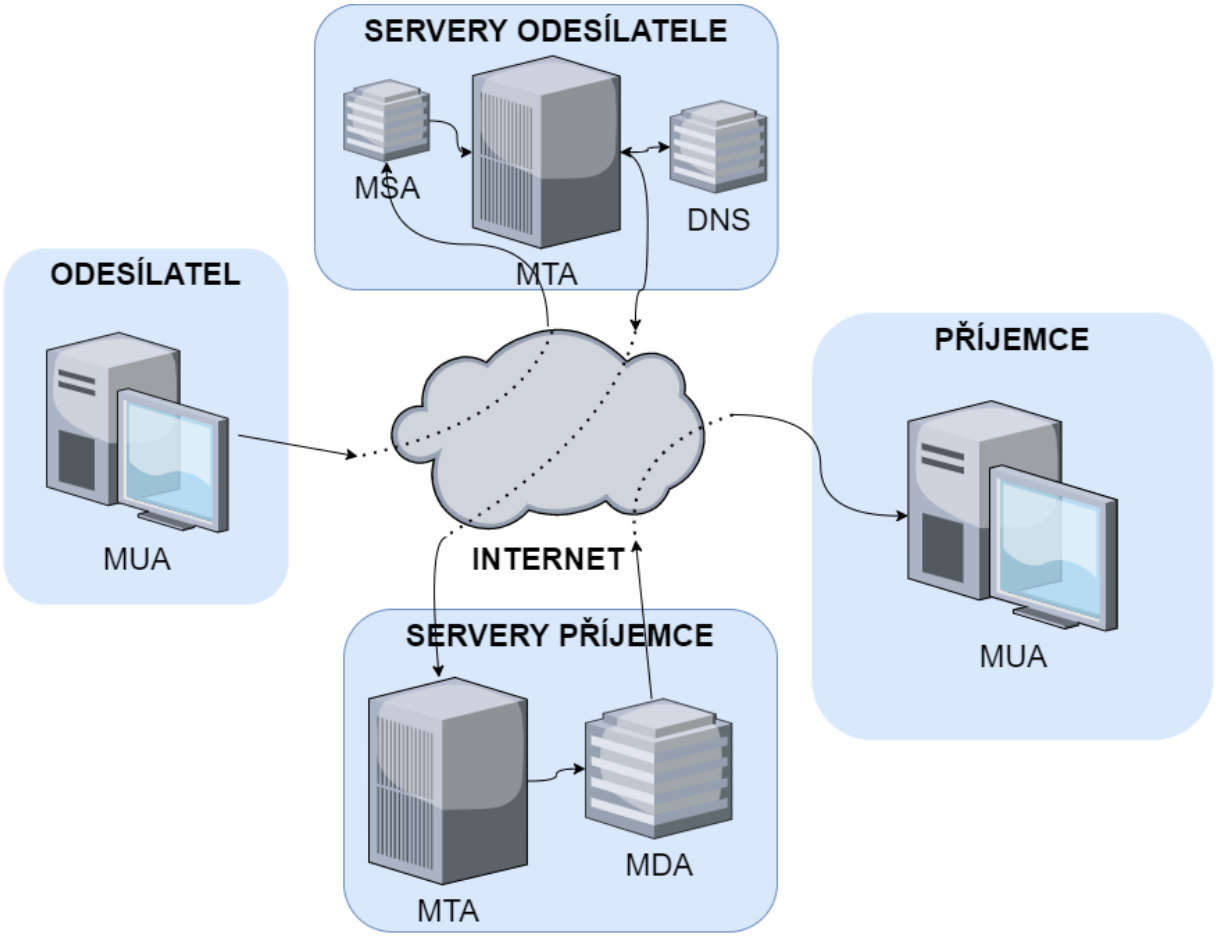
Struktura emailové komunikace se skládá ze softwarových i hardwarových komponent.

Z hardwarových komponent jsou to počítače a servery odesílatele a příjemce. Jak odesílací tak i přijímající servery jsou trvale připojeny k internetu. Klienti odesílatele i příjemce se k serverům připojují podle potřeby. Hardwarové subjekty podílející se na procesu doručování emailů se nazývají emailové uzly.[1]

Ze softwarových komponent jsou programy popř. programové moduly pro odesílání, přenos a příjem pošty - tzv. poštovní agenti.[2]

- Mail User Agent (MUA)
- Mail Submission Agent (MSA)
- Mail Transfer Agent (MTA)
- Mail Delivery Agent (MDA)
- Mail Retrieval Agent (MRA)

Kromě toho emailová infrastruktura využívá různé systémy a služby internetu jako například DNS (Domain Name System). [1]



Obrázek 2.1: Model přenosu emailové zprávy

2.1 Mail User Agent

Pokud se uživatel rozhodne zaslat email, musí pro tento úkon využít program, který umí zprostředkovat spojení se SMTP serverem. Tento software se nazývá Mail User Agent (MUA). Jedná se o emailového klienta s možností interakce s uživatelem.

Z uživatelského pohledu existují dva základní druhy těchto klientů. Každý tento klient má v sobě jednoho nebo více mailových agentů.

2.1.1 Emailový klient

Klasický emailový klient je desktopový program. Ten uživatelům umožňuje přístup k emailům na jejich počítači, aniž by se museli přihlašovat přes webový prohlížeč. Klienti jsou propojeni s emailovými účty prostřednictvím IP adresování. To znamená, že mohou pracovat i s emaily od poskytovatelů, kteří nenabízejí služby webové pošty (tzv. webmail). Stejně jako webmail mají klasičtí emailoví klienti přístup k adresám, funkcím chatu a

emailu, ale v pokročilejším měřítku. [3]

Přidanou výhodou je například možnost dodatečného šifrování a pokročilejšího zabezpečení. Nejznámější emailový klienti dnes navíc nabízí možnost rozšíření o tzv. zásuvné moduly (pluginy), které dále rozšiřují možnosti uživatelského nastavení pošty.

Všechny nové emaily přicházejí z předdefinovaného IMAP nebo POP3 serveru a jsou zpracovány agentem pro příjem pošty a uloženy na pevném disku počítače. Při odesílání emailové zprávy je zpráva odeslána emailový klientem na určený SMTP server prostřednictvím agenta pro odesílání pošty. [4]

Příkladem standardního emailového klienta je například Microsoft Outlook. Tato placená aplikace zahrnuje všechny výše popsané služby a navíc se integruje s aplikacemi Microsoft Office. Příkladem svobodného emailového klienta je Mozzila Thunderbird. Oba tyto příklady mají grafické rozhraní. Existují ale také konzoloví mailoví klienti. U nich se zpráva odesílá nebo přijímá za pomocí příkazů. Příkladem konzolového klienta je Mutt nebo Alpine.

2.1.2 Webmail

Druhým typem MUA je webový mailový klient. Webmail je emailový systém, ke kterému lze přistupovat prostřednictvím libovolného webového prohlížeče. Podmínkou je připojení k internetu. Všechny emaily, kalendářové služby a kontakty musí být hostovány na serverech poskytovatele. [3] [4]

Většina systémů webové pošty je zdarma. Jsou tedy ideální pro jednotlivce a malé společnosti, které si nemohou dovolit nebo nechtějí platit za oficiální emailovou službu. Některé z nejoblíbenějších bezplatných možností jsou Gmail od Googlu, Yahoo Mail nebo pro českou lokalizaci Email od společnosti Seznam.cz. [5]

2.2 Mail Retrieval Agents

Agent pro načítání pošty (MRA) je počítačová aplikace, která načítá nebo získává emaily ze vzdáleného poštovního serveru a spolupracuje s agentem pro doručování pošty (MDA) při doručování pošty do místní nebo vzdálené emailové schránky. [6]

MRA se často využívá pro přístup k sekundárním poštovním schránkám, které mohou mít uživatelé na různých serverech. Tyto sekundární schránky se pak spojí s primární poštovní schránkou. [2]

MRA mohou být samy o sobě externí aplikace, většinou ale bývají součástí větší aplikace, jako je např. MUA. Mezi významné příklady samostatných MRA patří Fetchmail a Getmail.

Koncept MRA není v architektuře elektronické pošty standardizovaný. Přestože fungují jako agenti pro přenos pošty (MTA), jsou MRA při načítání a odesílání zpráv technicky klienty. [6]

2.3 Mail Transfer Agent

Emailový server má mnoho názvů, ale odbornou veřejností nejpoužívanější je název Mail Transfer Agent (MTA). MTA se starají o směrování a přenos pošty. V emailové komunikaci může zpráva projít přes jednoho nebo i více těchto agentů. Ti mezi sebou komunikují prostřednictvím protokolu SMTP. [7]

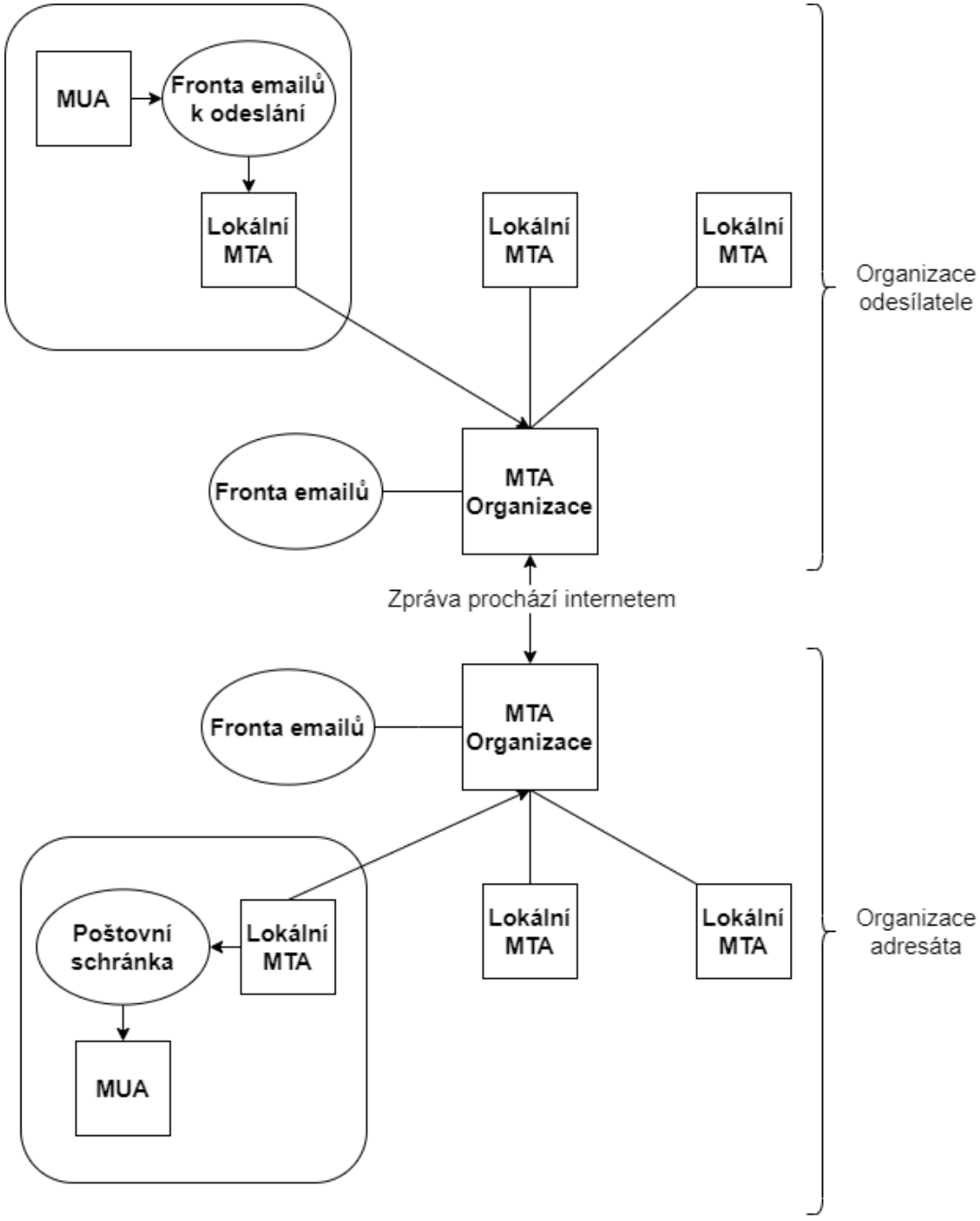
MTA obsahuje mnoho funkcionalit, aby mohl zajišťovat přenos pošty. Mezi základní funkce patří:

- rozumět formátům emailového adresování
- řazení do fronty a řízení přeposílání
- zpracovávat vrácené emaily a další chybové stavy
- starat se o přeposílání, aliasy atd.

U emailových serverů je také velmi důležité, aby implementovaly určité bezpečnostní mechanismy, aby se zabránilo zneužití. Na vstupu MTA komunikují s MUA, ale dnes se častěji přidává mezičlánek do komunikace — tzv. Mail Submission Agent. Příklady MTA jsou Sendmail nebo Postfix. [2] [8]

2.3.1 Směrovní elektronické pošty

Jedním z nejpodstatnějších vlastností elektronické pošty je možnost předávání zprávy mezi emailovými servery. To umožňuje zaslat zprávu uživateli, který má svojí emailovou schránku u zcela jiného poskytovatele. Tato vlastnost s sebou nese ale také spoustu komplikací jako je např. bezpečnost nebo zajištění doručení do cílové destinace. [9]



Obrázek 2.2: Příklad směrování elektronické pošty

Proto je jedním z nejdůležitějších úkolů MTA správné nasměrování zprávy. Zaslaná pošta může projít několika uzly (servery neboli MTA) směrem k centrálnímu poštovnímu serveru odesílatele. Poté je přenesen internetem na centrální server doručovatele a odtud je předávána jeho subsystémy až do cílové destinace na SMTP server, který má za úkol konečné doručení pošty adresátovi. [9]

Pokud jsou obě emailové adresy ze stejné emailové domény, zpráva je poslána okamžitě.

2.3.2 Mail Submission Agent (MSA)

Jedná se o program běžící zpravidla na emailovém serveru odesílatele, který je určený pro příjem pošty z MUA. Ne všichni emailoví klienti jsou totiž schopni vložit email do systému korektním způsobem.

Jelikož komunikuje přímo s autorským MUA, může opravit drobné chyby ve formátu zprávy (například chybějící údaje jako datum nebo adresu s chybějícím názvem domény) popřípadě ihned nahlásit chybu autorovi, aby ji mohl opravit před odesláním některému z příjemců.

MTA, který přijímá zprávu z jiného webu, nemůže spolehlivě provést tyto opravy a jakákoli chybová hlášení vygenerovaná takovým MTA se k autorovi dostanou (pokud vůbec) až poté, co je zpráva již odeslána. [10] [2]

2.4 Mail Delivery Agents

Tento programový modul běží na serveru. Stará se o přijímání pošty ze SMTP serveru (MTA) a ukládá ji do poštovní schránky uživatele. MDA je také znám pod zkratkou LDA (Local Delivery Agent). [11]

Když email dosáhne svého konečného cíle, měl by být vložen do poštovní schránky uživatele (často nazývané „INBOX“). Tradičně se jednalo o prostý soubor ve formátu mbox, ale úložiště poštovní schránky lze implementovat i jiným způsobem — např. jako poštovní databáze. Je na doručovacím agentovi, aby provedl úkol uložit příchozí poštu na trvalé místo pro následný přístup uživatele.[2]

Příkladem je server Dovecot, který je především serverem POP3 a IMAP umožňujícím MUA načítat poštu. Obsahuje ale také MDA, který přijímá poštu z MTA a doručuje ji do poštovní schránky serveru. [11]

2.4.1 Mail Access Agents

O doručení zprávy se stará MDA, který čeká, až příjemce přistoupí k jeho emailu. MUA příjemce většinou nemá přímý přístup k úložišti pošty a kontaktuje agenta pro přístup k poště (MAA), aby mohl s úložištěm zpráv pracovat jeho jménem.

Úkolem MAA neboli agenta pro přístup ke zprávám je přenést zprávu ze schránky na poštovním serveru na straně příjemce do počítače příjemce, agent pro přístup ke zprávám je také konečným doručovatelem na straně příjemce. MAA komunikuje s MUA pomocí přístupového protokolu, jako je POP3 nebo IMAP. [12]

Příklady MAA jsou Dovecot a Qpopper.

Kapitola 3

Emailové komunikační protokoly

Emailová infrastruktura má mnoho protokolů, z nichž každý plní specifickou funkci v procesu odesílání nebo přijímání emailových zpráv. Běžně používané protokoly emailového systému jsou SMTP pro odesílání pošty a protokoly POP3 nebo IMAP pro její příjem.

Po ustanovení TCP spojení mezi klientem a serverem, odesílá klient emailovou zprávu na server pomocí protokolu SMTP. Pomocí stejného protokolu dochází k přeposílání pošty mezi jednotlivými servery. Pro stažení zprávy ze serveru příjemce na počítač se používá protokol POP3 nebo IMAP. Procesy odesílání a získávání emailů probíhají prostřednictvím klient/server procedury.[13]

3.1 Formát internetových zpráv

Aby byly digitální informace obsažené v jakémkoli emailu přenášeny mezi servery (MTA), musí být dodržena pravidla stanovená v několika standardech. Jakákoliv pošta, která nevyhovuje, je vystavena riziku, že bude odmítnuta přísnějším serverem (MTA).

K protokolu SMTP dle normy RFC 2821 [14] existuje doprovodný dokument RFC 2822 [15] s názvem „Internet message format“ (Formát internetových zpráv). Tato norma specifikuje syntaxi textových zpráv, které jsou posílány mezi uživateli počítačů v rámci elektronické pošty. [15] V SMTP protokolu se jedná o fázi SMTP DATA. [2]

První část zprávy se skládá z několika řádků záhlaví pošty, za nimiž následuje prázdný řádek, za nímž následuje tělo zprávy. Řádky se skládají ze znaků US-ASCII (1-127) a jsou ukončeny kombinací znaků „CRLF“. To jsou řídicí znaky, které se používají k označení zlomu řádku v textovém souboru. Řádky mohou obsahovat až 998 znaků. [15]

3.1.1 Hlavičky emailu

Každý řádek záhlaví se skládá z názvu pole (ze znaků ASCII (33-126)), dvojtečky (‚:’), znaku mezery (, ’) a těla pole v tomto pořadí.

Každé pole záhlaví je logicky tvořeno jedním řádkem znaků, který obsahuje název pole, dvojtečka a tělo pole. Pro větší pohodlí a kvůli omezení 998 znaků na řádek, lze část těla pole záhlaví rozdělit na více řádků — tzv. „skládání“. Obecným pravidlem je, že všude tam, kde tato norma umožňuje zalomení bílého místa, lze vložit znak CRLF. Tělo zprávy má v podstatě volný formát, i když specifikace MIME zavádí nové podformáty. [15]

Hlavičky pošty obsahují důležité meta informace o obsahu zprávy. Obsahují informace, jako je: [15]

- datum vytvoření pošty (Date:)
- pole původních majitelů jako např. autor (From:) a skutečný odesílatel (Sender:)
- pole cílové adresy jako je příjemce (To:) a příjemci kopie (Cc:)
- identifikační pole jako je jedinečný identifikátor zprávy (Message-ID:)
- odkazující informace (In-ReplyTo:, References:)
- informační pole jako předmět (Subject:)
- pole pro sledování, která obsahují části obálky SMTP, jako jsou zprostředkující MTA (Received:) a zpáteční adresa odesílatele (Return-Path:)

Dále pak také obsahuje celou řadu (nepovinných) polí přidaných MUA nebo filtrovacím softwarem. [16]

3.1.2 MIME

MIME (Multipurpose Internet Mail Extensions) v překladu znamená — víceúčelová rozšíření internetové pošty. Důvodem vzniku této normy, respektive 5 spolu souvisejících norem, bylo vyřešení problémů se SMTP standardem. Ten umožňoval pouze základní 7bitové ASCII znaky uvnitř hlaviček a těl emailů. S vývojem internetu rostla poptávka po podpoře mezinárodních znakových sad a přenosu multimediálního obsahu v rámci emailu. [17]

MIME zavádí speciální hlavičku „Content-Type:“, aby bylo možné specifikovat typ MIME a podtyp MIME následujícího obsahu. Příklady typů/podtypů jsou text/plain a image/jpeg. [18]

Aby bylo možné pracovat v mezích SMTP protokolu, přidává MIME do zprávy hlavičku kódování přenosu obsahu (Content-Transfer-Encoding:). Nejpoužívanější kódování pro přenos libovolných binárních dat jsou „quoted printable“ a „base64“. [19]

Base64 je nejvhodnější pro libovolná binární data, kde jsou každé 3 bajty reprezentovány 4 bajty v tisknutelném rozsahu ASCII. Quoted printable je vhodnější pro texty, kde se občas vyskytne tzv. vysoký ASCII bajtem. (ASCII tabulka je definována pro znaky 0-127 bitů). [2]

MIME zavádí speciální syntaxi tzv. kódovaného slova („encoded word“) pro použití v polích záhlaví. Obecně je kódované slovo posloupnost znaků ASCII, která začíná znakem „=?“, končí znakem „?=“. Mezi nimi jsou dva znaky „?“. Písmeno uvnitř určuje metodu kódování.

Příklad záznamu kódovaného slova — Subject: =?iso-8859-1?Q?=A1Hola,_se=F1or!?= znamená „Subject: ¡Hola, señor!“.

Písmeno ‚Q‘ (?Q?) říká, že pro kódování bylo použito kódování quoted-printable. (V případě použití base64 kódování by místo ‚Q‘ bylo písmeno ‚B‘.)

V emailu založeném na MIME je nutné zadat znakovou sadu ve formátu typ/podtyp (příklad — text/plain) pomocí parametru charset (character set — znaková sada). Výchozí znaková sada je US-ASCII, ale někdy MUA znakovou sadu nezadávají, i když se vyskytují jiné znaky než US-ASCII, nebo dokonce zadávají špatnou znakovou sadu. Navíc webové MUA (webmail) má často potíže s určením znakové sady, kterou má použít. [2][18][19][20][21][22]

3.1.3 Formát úložiště pošty

Formát samotné zprávy, se v zásadě liší od způsobu, jakým jsou poštovní zprávy ukládány do trvalého nebo dočasného úložiště.

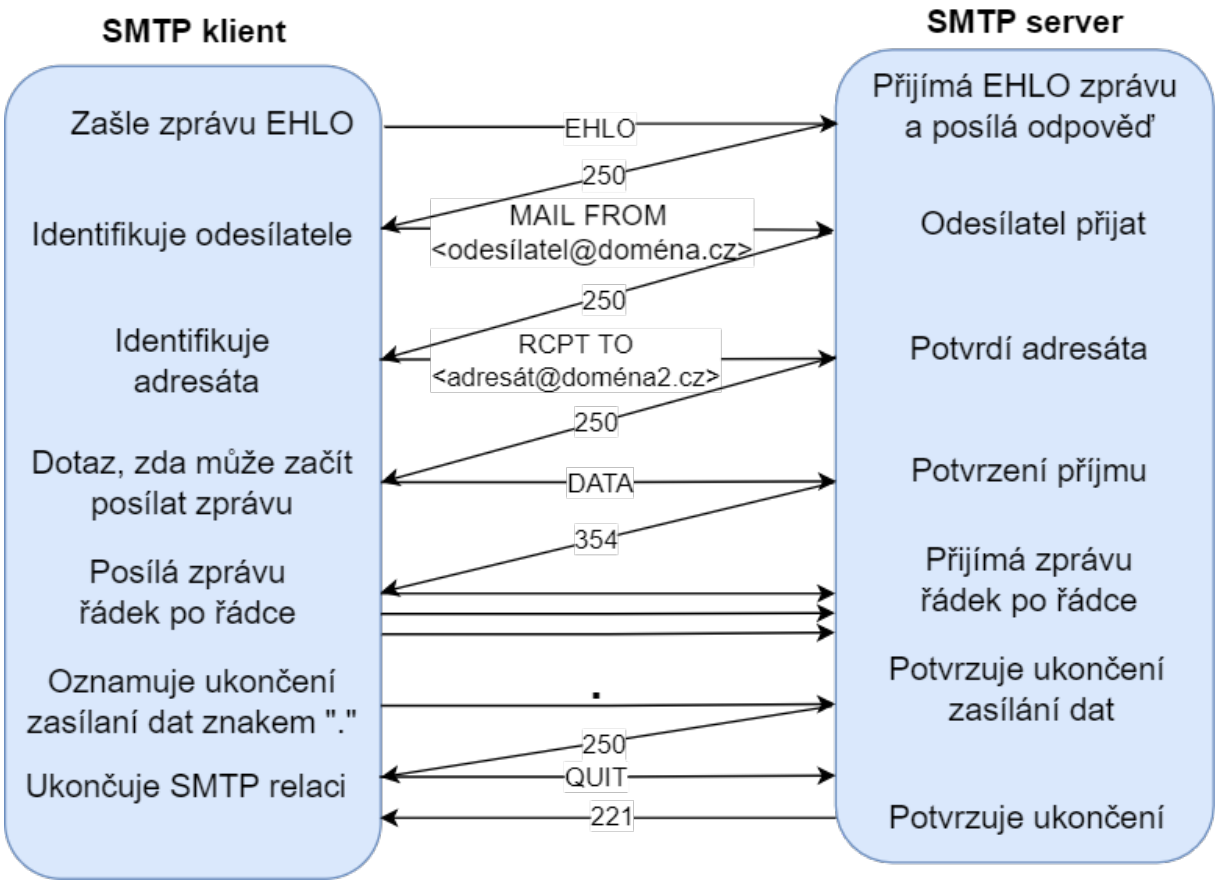
Pošta může být uložena v plochých souborech, případně s místními konvencemi pro oddělování konců řádků a speciálními oddělovacími řádky jako v klasickém příkladu formátu „mbox“.

Pošta může být také umístěna v adresářích s jedním souborem na jednu poštovní zprávu, jako je tomu u MH složek (formát pro ukládání zpráv) nebo modernějšího a robustnějšího Maildiru, který používá poštovní program qmail. Pošta může být také uložena v databázovém systému, jako to dělá software Cyrus. [2]

3.2 SMTP

SMTP je protokol aplikační vrstvy pro přenos elektronické pošty mezi různými hostiteli v rámci TCP/IP komunikačního modelu. Klient, který chce odeslat poštu, otevře TCP spojení k SMTP serveru. Poté se pomocí tohoto spojení pokusí odeslat poštu. Server standardně poslouchá na TCP portu 25. Na tomto portu pak klient zahájí spojení a následný přenos dat.

Když je TCP spojení úspěšné, dojde k sekvenci jednoduchých dialogů požadavek–odpověď definovaných protokolem SMTP, ve kterém klientský proces přenese poštovní adresy odesílatele a příjemce (příjemců) zprávy. [23]



Obrázek 3.1: Příklad SMTP komunikace klient — server

Formát emailové adresy

Ve specifikaci RFC5321 [24] se adresou rozumí řetězec znaků, který identifikuje uživatele, jemuž bude pošta odeslána, nebo místo, do něhož bude pošta odeslána. kde bude pošta uložena. [24]

Nejstarším formátem emailové adresy byl formát používaný v síti ARPANET — user@host. Tento formát předpokládá tzv. plochý jmenný prostor, což znamená, že každý uživatel sítě má jedinečné jméno. Zpočátku byl tento formát dostatečný, ale s rychlým vývojem internetu ne dlouho udržitelný. [2]

Řešení problému přišlo s vynalezením DNS (Domain Name System) v roce 1983. DNS nabízí distribuovanou databázi. To umožňuje správu jednotlivých podprostorů delegovat na autonomní servery. Tyto servery pak mají nad daným podprostorem pravomoc. [2]

Označení používaná pro jednotlivé vrstvy hierarchie jsou oddělena tečkami. Každá

vrstva musí pouze zkontrolovat, zda jsou štítky používané pro její vlastní podvrstvy jedinečné.

Emailové adresy mají nyní podobu user@host.some.domain, kde je za znakem @ povolen libovolný počet vrstev.

Podstatným krokem vpřed pro DNS bylo přidání rozšíření tzv. MX záznamu („Mail Exchanger“). Do této doby pro směrování a doručování pošty vždy používal mechanismus ukládání a předávání. Tudíž nebylo nutné přímé spojení mezi počítači, které si vyměňovaly poštu. Naproti tomu MX záznamy poskytují speciální vrstvu pro emailové adresy. Například pošta na adresu user@some.domain by mohla být směrována do další destinace prostřednictvím MX záznamu — například na adresu relay.some.domain. [2]

example.com	record type:	priority:	value:	TTL
@	MX	10	mailhost1.example.com	45000
@	MX	10	mailhost2.example.com	45000

Obrázek 3.2: Příklad MX záznamu na DNS serveru [25]

Pro směrování poštovní zprávy je rozhodující pouze doménová část (za znakem „@“) emailové adresy příjemce. Tato doménová část se vyhledá v DNS, aby se zjistilo, zda je pro ni definován MX záznam. Pokud je záznam definován, hostitelé pošty se následně pokusí směrovat poštu v pořadí na základě přidělené priority. Pokud není nalezen žádný MX záznam, pošta by měla být doručována na IP adresu spojenou s doménovou částí pomocí A záznamu. A záznam je základním typem doménového záznamu. Jedná se záznam, který spojuje název domény s IP adresou. [26]

Tento způsob je však pouze základní myšlenka doručování pošty. MTA standardně mají nastavená vlastní pravidla, které tento přístup nahrazují.[2]

example.com	record type:	value:	TTL
@	A	192.0.2.1	14400

Obrázek 3.3: Příklad A záznamu na DNS serveru [27]

3.2.1 Základní příkazy SMTP

Příkazy SMTP dle normy RFC 2821 [14] definují přenos pošty nebo funkci poštovního systému požadovanou uživatelem. Příkazy SMTP jsou znakové řetězce. Základní syntaxe

příkazu je: *<Kód-příkazu><parametry>*. Znakový řetězec je ukončen buď znakem *<SP>* (pokud následují další parametry), nebo znakem *<CRLF>* (nenásleduje žádný další parametr). [24]

Poštovní transakce zahrnuje několik datových objektů, které jsou předávány jako argumenty různým příkazům. Tyto datové objekty jsou přenášeny a jsou udržovány v čekání, dokud není sděleno potvrzení ukončením indikace dat pošty, která dokončí poštovní transakci. [28]

K dispozici jsou různé vyrovnávací paměti (buffery) pro uložení různých typů datových objektů. Specifické příkazy způsobí připojení informací ke konkrétní vyrovnávací paměti nebo vymazání jednoho nebo více vyrovnávacích pamětí. [29]

Existují i příkazy, které vyžadují speciální zadání parametrů. Dále rovněž existuje mnoho rozšíření základního SMTP serveru. Tyto rozšíření lze povolit nastavením relace pomocí příkazu EHLO a příslušných kódů pro rozšíření. Důležitou podmínkou je, že oba SMTP servery musí podporovat dané rozšíření. Tato část stručně popisuje různé příkazy a rozšíření SMTP

V uvedených příkladech komunikace SMTP se písmena C a S používají k označení příkazů vydávaných klientovi (odesílateli) a odpovědí zaslaných serverem (příjemcem). [29]

HELO a EHLO

Tyto příkazy slouží pro zahájení celého procesu odesílání emailů. Klient odešle příkaz HELO nebo EHLO na server SMTP, aby se identifikoval a zahájil konverzaci. Jako argument pro tyto příkazy je odeslán plně kvalifikovaný název domény SMTP klienta. To znamená, že definuje absolutní cestu k uživateli dle DNS standardů.

Tyto příkazy a odpověď „250 OK“ na jeden z nich potvrzují, že klient SMTP i server SMTP jsou v počátečním stavu. To znamená, že neprobíhá žádná transakce a všechny tabulky stavů a buffery jsou vymazány. Syntaxe těchto příkazů je HELO NázevDomény nebo EHLO NázevDomény.[30]

Příklad Příkazu HELO: [29]
C: HELO mojedomena.cz
S: 250- mailboxXXXX.mailhostingXXXX.com

Norma RFC 821 byla první, která definovala příkaz HELO jako uvítací příkaz SMTP, což bylo provedeno v dobách, kdy byly poštovní transakce považovány za bezpečné a jednoduché bez jakýchkoli dalších protokolů, jako např. TLS. Po poslední revizi RFC 2821, který definuje nový SMTP jako ESMTP, přidává nové funkce, a to jak do protokolu, tak

i do přenosových mechanismů. Nicméně aby toto fungovalo, SMTP klient (server odesílající poštu) potřebuje mechanismus, který ověří, zda je SMTP server schopen tyto metody podporovat. Jediný způsob, jak zjistit možnosti protější strany, je po pozdravu a těsně před začátkem přenosu.[14] [30]

Příklad Příkazu EHLO: [29]
C: EHLO mojedomena.cz
S: 250- mailboxXXXX.mailhostingXXXX.com
S: 250-PIPELINING
S: 250-SIZE 20971520
S: 250-VRFY
S: 250-ETRN
S: 250-AUTH PLAIN LOGIN
S: 250-AUTH=PLAIN LOGIN
S: 250-ENHANCEDSTATUSCODES
S: 250-8BITMIME
S: 250-DNS

Původní příkaz HELO neobsahoval výše zobrazená data. Tato data jsou důležitá pro komunikaci a proto bylo nutné jejich začlenění do systému. Na základě tohoto požadavku vznikl nový příkaz, který data obsahuje. To je důvod, proč existují dva příkazy. Od RFC 2821 je vyžadováno všemi klienty SMTP, aby používali právě příkaz EHLO místo příkazu HELO. Oba příkazy by ale měly být stále přijímány jakýmkoli SMTP serverem. [14]

MAIL FROM

Tento příkaz se používá k zahájení poštovní transakce, ve které jsou data doručena SMTP serveru. Od této chvíle je server připraven přijmout emailovou adresu. Server může data doručit do jedné nebo více poštovních schránek. Stejně tak může data předat jinému serveru. Po přijetí dat server reaguje kódem odpovědi 250 OK.

Příkaz obsahuje jako argument cestu k původnímu odesílateli. To jest jméno odesílatele, ale může to být také seznam hostitelů, kteří byli použiti k předání poštovní zprávy od původního odesílatele. V seznamu hostitelů je prvním hostitelem aktuální přijímající server SMTP. Poslední je cíl emailu. [30]

Syntaxe tohoto příkazu je MAIL FROM: (<>/Reverse-Path) [Parametry emailu]. [24]

RCPT TO (Recipient To)

Další příkaz následuje po kódu odpovědi 250 OK, který identifikuje, komu je email odeslán. Server SMTP opět odpoví stejným kódem a v tomto okamžiku lze odeslat další příkaz RCPT TO s emailovou adresou jiného příjemce.Tato akce proběhne právě tolikrát, kolik uživatelů má danou zprávu obdržet.

Pole argumentů obsahuje dopřednou cestu a může obsahovat volitelné parametry. Pokud bylo sjednáno rozšíření služby, příkaz RCPT může také nést parametry spojené s konkrétním rozšířením nabízeným serverem. [31]

Syntaxe tohoto příkazu je RCPT TO: (<Postmaster@ doména>/ <Postmaster>/ Forward Path) [Rcpt-parameters]. [24]

DATA

Tento příkaz odešle klient v okamžiku, kdy chce zahájit přenos zprávy. Na základě tohoto příkazu dojde k připojení do bufferu emailových dat. Následuje přenos skutečných dat, která tvoří emailovou zprávu. To zahrnuje jak hlavní text, tak záhlaví, jako je předmět. Přenos dat je ukončen řádkem obsahujícím pouze tečku, tedy posloupnost znaků <CRLF>.<CRLF>.

Veškerý obsah zprávy se přesune na server SMTP, který odpoví kódem 345. Pokud je přijat a připraven k doručení, server odešle další 250 OK kód. V tomto okamžiku je zpráva na cestě k příjemcům. [29]

Přijetí indikace dat konce pošty vyžaduje, aby server zpracoval uložené informace o poštovní transakci. Toto zpracování spotřebovává informace ve vyrovnávací paměti zpětné cesty, vyrovnávací paměti dopředné cesty a vyrovnávací paměti dat pošty a po dokončení tohoto příkazu jsou tyto vyrovnávací paměti vymazány.

Pokud je zpracování úspěšné, musí příjemce odeslat OK odpověď. Když server SMTP přijme zprávu buď pro přenos nebo pro konečné doručení, vloží záznam sledování, který se také nazývá řádek časového razítka nebo řádek přijatých zpráv, na začátek dat pošty.

Syntaxe tohoto příkazu: DATA (následuje tok dat) CRLF. [14]

VRFY (Verify)

Tento příkaz požádá server o potvrzení, že zadané uživatelské jméno nebo poštovní schránka jsou platné (existují). Pokud je dotazováno uživatelské jméno, je vráceno celé jméno uživatele a plně zadaná poštovní schránka. U některých emailových serverů je příkaz VRFY ignorován, protože může představovat bezpečnostní díru. Příkaz lze použít k prozkoumání přihlašovacích jmen na serverech. Servery, které příkaz VRFY ignorují, obvykle odešlou nějakou odpověď, ale neodešlou informace, které klient požadoval. [30]

Syntaxe tohoto příkazu je: VRFY <uživatelské jméno>. [24]

Příklad Příkazu VRFY: [29]
C: VRFY bob@b.com
S: 252 2.0.0bob@b.com
C: VRFY bob@c.com
S: 554 5.7.1 <bob@c.com>:Relay access denied

AUTH

Příkaz AUTH je rozšíření služby SMTP, pomocí kterého může klient SMTP indikovat autentizační mechanismus serveru, provést výměnu autentizačního protokolu, vyjednat vrstvu zabezpečení pro následné interakce protokolu během této relace a během poštovní transakce.

Toto rozšíření obsahuje SASL (Simple Authentication and Security Layer) profil pro SMTP. Klíčové slovo AUTH EHLO obsahuje jako parametr seznam dostupných SASL mechanismů. Seznam dostupných mechanismů se může změnit po příkazu STARTTLS.

Syntaxe tohoto příkazu je: AUTH mechanismus. Mechanismus je seznam identifikující SASL autentizační mechanismus, kterým může být PLAIN, LOGIN, CRAM-MD5, DIGEST-MD5 atd. [32]

Autentizační mechanismus vybírá, jak se přihlásit a jakou úroveň zabezpečení má použít. Počáteční odpověď je volitelná počáteční odpověď klienta, která musí být zakódována v base64 nebo obsahovat jeden znak ‚=‘.

V případě, že je použito klíčové slovo PLAIN, uživatelské jméno a heslo nejsou odeslány jako prostý text. Jsou zakódovány pomocí kódování base64. Je také možné odeslat uživatelské jméno a heslo spolu s příkazem AUTH PLAIN jako jeden řádek.[32]

LOGIN mechanismus je další běžnou metodou přihlášení k serveru SMTP. Níže uvedený příklad komunikace SMTP ukazuje, jak lze AUTH LOGIN použít k ověření přihlášení k serveru:

Ukázka použití AUTH LOGIN: [29]

C: EHLO MYDOMAIN.COM
S: 250- mailboxXXXX.mailhostingXXXX.com
S: 250-PIPELINING
S: 250-SIZE 20971520
S: 250-VRFY
S: 250-ETRN
S: 250-AUTH PLAIN LOGIN
S: 250-AUTH=PLAIN LOGIN
S: 250-ENHANCEDSTATUSCODES
S: 250-8BITMIME
S: 250-DNS
C: AUTH LOGIN
S: 334 VXNlcm5hbWU6
C: dEBuaWN0c29mdC5jb20=
S: 334 UGFzc3dvcmQ6
C: dGVzdA==
S: 235 2.7.0 Authentication successful

Ve výše uvedeném příkladu je odpověď na AUTH LOGIN *„334 VXNlcm5hbWU6“*, což je stavový kód 334 s kódovanou zprávou base64 „Username:“. Uživatelské jméno zakódované v base64 *„dEBuaWN0c29mdC5jb20=“* z textového uživatelského jména *t@nictsoft.com* je uvedeno v další odpovědi od klienta. Server odpoví *„334 UGFzc3dvcmQ6“*, což je stavový kód 334 s kódovanou zprávou base64 „Heslo:“. Base64 kódované heslo *„dGVzdA==“* je uvedeno v další odpovědi od klienta. Poté je autentizace úspěšná. [29]

Nevýhodou použití autentizačních mechanismů PLAIN a LOGIN je, že uživatelské jméno a heslo lze dekódovat. Pro dosažení vyšší bezpečnosti lze použít autentizační mechanismus CRAM-MD5.

CRAM-MD5 kombinuje autentizační mechanismus výzva-odpověď pro výměnu informací a kryptografický algoritmus Message Digest 5 (MD5) pro šifrování důležitých informací. [32]

Poté, co je příkaz AUTH CRAM-MD5 odeslán na server, server odešle zpět klientovi jednorázovou výzvu v base64 formátu. Klient odpoví zasláním řetězce ve stejném formátu serveru. Odeslaný řetězec obsahuje uživatelské jméno a 16bajtový hash v hexadecimálním zápisu.

Hash je výstupem výpočtu HMAC (viz. kapitola TLS). Heslo je použito jako tajný klíč a původní výzva serveru jako zpráva. Server také vypočítá svůj vlastní hash se svou představou hesla uživatele, a pokud se hash klienta a hash serveru shodují, pak byla autentizace úspěšná. Klientovi je pak zaslán kód odpovědi 235. [32] [29] [30] [31]

STARTTLS

Emailové servery a klienti používající protokol SMTP obvykle komunikují prostřednictvím internetu pomocí prostého textu. Komunikace často probíhá přes jeden nebo více entit, které nejsou kontrolovány nebo důvěryhodné pro server a klienta. Tuto komunikaci lze tedy číst nebo hůř — měnit samotný obsah.

Pro zvýšení bezpečnosti lze při komunikaci mezi emailovým serverem a klientem použít šifrované spojení TLS (Transport Layer Security). Protokol TLS je nejužitečnější v případě, kdy je třeba zašifrovat přihlašovací uživatelské jméno a heslo (odesílané příkazem AUTH). TLS lze použít k šifrování celé emailové zprávy, ale příkaz nezaručuje, že celá zpráva zůstane zašifrovaná po celou cestu k příjemci. Některé emailové servery se mohou rozhodnout odeslat emailovou zprávu bez šifrování. Ale alespoň uživatelské jméno a heslo použité příkazem AUTH zůstanou zašifrované. Použití příkazu STARTTLS spolu s příkazem AUTH je velmi bezpečný způsob ověřování uživatelů. [30]

QUIT

Když byl email odeslán, klient odešle příkaz QUIT na server, čímž přeruší spojení. Pokud byl úspěšně uzavřen, server odpoví kódem 221. [14]

RSET (Reset)

Tento příkaz je odeslán na server, když je třeba zrušit poštovní transakci. Neuzavře připojení, ale vše resetuje a odstraní všechna předchozí data o emailu a zúčastněných stranách. Toto běžně použijete, když dojde k chybě, jako je zadání nesprávných informací o příjemci, a proces je třeba restartovat. [14]

3.3 POP3

Protokol Post Office Protocol (POP3) je standardní internetový protokol aplikační vrstvy. Je používán mailovými klienty (MUA) k načítání emailů ze vzdáleného poštovního serveru prostřednictvím TCP/IP připojení. [33]

Emailoví klienti obvykle používají k připojení k serveru POP3 TCP port 110. Pokud je na serveru POP3 podporována šifrovaná komunikace, může si uživatelé volitelně vybrat,

zda se připojí pomocí příkazu STLS po iniciační fázi protokolu, nebo pomocí protokolu POP3S. Ten používá pro připojení k serveru protokol TLS nebo SSL na portu TCP 995. [34]

Jakmile je email u klienta, POP3 jej odstraní ze serveru. U některých implementací mohou uživatelé nebo správce určit, že pošta zůstane nějakou dobu uložena na serveru, což uživatelům umožní stahovat emaily tolikrát, kolikrát si přejí během zadaného období. [33]

3.3.1 Jak funguje

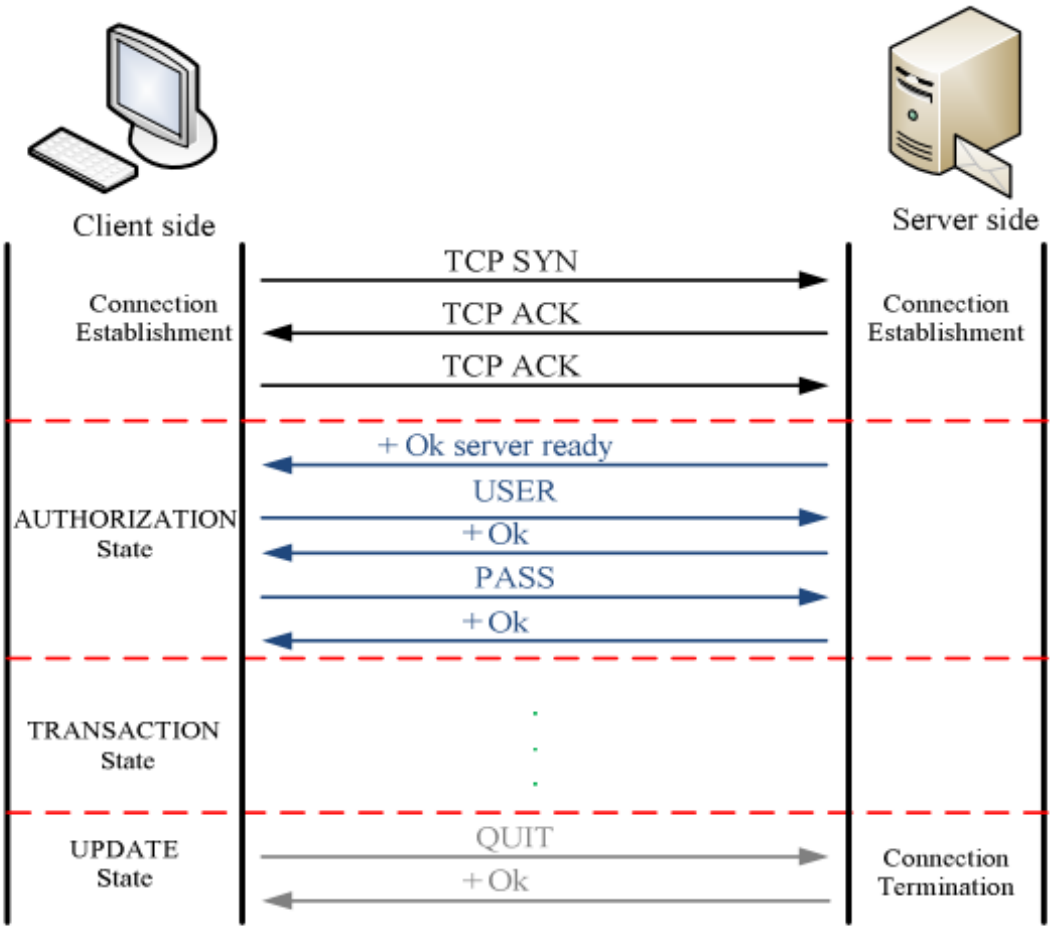
Proces POP3 klient/server začíná, když má emailový klient příchozí emailovou zprávu a chce ji získat ze serveru. Postup začíná, když klient naváže nové TCP spojení mezi klientem a emailovým serverem na portu 110. Toto spojení slouží k přenosu řídicích příkazů a dat emailové zprávy do poštovní schránky emailového klienta. [34]

Jakmile je navázáno TCP spojení, klient čeká na potvrzení od emailového serveru, obsahující uvítací zprávu +OK a informuje klienta, že server je připraven zahájit mezi nimi příkazové dialogy. Klient nyní projde první relací, která se nazývá AUTHORIZATION State (Stav AUTORIZACE). [35]

V této fázi se klient musí identifikovat na emailovém serveru pomocí příkazu USER, který obsahuje uživatelské jméno. Pokud je uživatelské jméno odeslané na emailový server správné, server odpoví příkazem +OK jako potvrzení. Pokud je uživatelské jméno odmítnuto, server odešle příkaz -ERR. Další krok v tomto stavu je vydání hesla uživatelského jména, které se provádí pomocí příkazu PASS. [35]

Pokud je heslo správné, server odpoví příkazem +OK, v opačném případě server vydá příkaz –ERR, takže klient musí heslo serveru odeslat znovu. Jakmile jsou příkazy USER a PASS provedeny s kladnými odpověďmi z emailového serveru, je klientovi poskytnut přístup k příslušné poštovní schránce. [35]

Klient poté vstoupí do tzv. TRANSACTION State (Stav TRANSAKCE) — spustí proces dialogu příkazů, dokud klient nedokončí proces načítání emailových zpráv ze serveru. Pokud již nejsou k dispozici žádné další emailové zprávy k načtení, klient provede příkaz QUIT, aby tento stav opustil. V této fázi POP3 server, pokud není nastaveno jinak, uvolní schránku na serveru a uzavře TCP spojení. [13] [35]



Obrázek 3.4: POP3 komunikace [13]

3.3.2 Výhody POP3 protokolu

- Umožňuje uživatelům číst emaily offline. Vyžaduje připojení k internetu pouze v době stahování emailů ze serveru. Jakmile jsou emaily staženy ze serveru, všechny stažené emaily jsou umístěny na pevném disku počítače.
- Vyžaduje méně úložného prostoru na serveru, protože všechny emaily jsou uloženy na místním počítači.
- Neexistuje žádné omezení velikosti emailu, který přijímáme nebo odesíláme.

3.3.3 Nevýhody POP3 protokolu

- Pokud jsou emaily staženy ze serveru, jsou všechny, ve výchozím nastavení, odstraněny ze serveru. Takže k emailům nelze přistupovat z jiných počítačů, pokud nejsou

nakonfigurovány tak, aby ponechaly kopii pošty na serveru.

- Protože všechny přílohy jsou uloženy na lokálním počítači, existuje riziko napadení virem ukrývajícího se např. v nějaké příloze emailu. Je na antivirové službě, aby lokální schránku pravidelně kontrolovala.

3.4 IMAP

Internet Message Access Protocol (IMAP) je standardní protokol aplikační vrstvy používaný emailovým klientem pro přístup k emailům uloženým na vzdáleném poštovním serveru.

Emailoví klienti, kteří používají protokol IMAP, mohou použít buď port 143 pro navázání nezabezpečeného připojení, nebo port 993. Pokud vyžadují bezpečné připojení prostřednictvím protokolu IMAP přes protokol SSL/TLS (IMAPS).[36]

Protokol IMAP prošel několika revizemi a jeho poslední verze IMAP4 revize 2 byla definována v normě RFC 9051 v roce 2021. [37]

3.4.1 Jak funguje

Procedura klient/server protokolu IMAP4 začíná navázáním spojení TCP mezi klientem a emailovým serverem na portu 143.

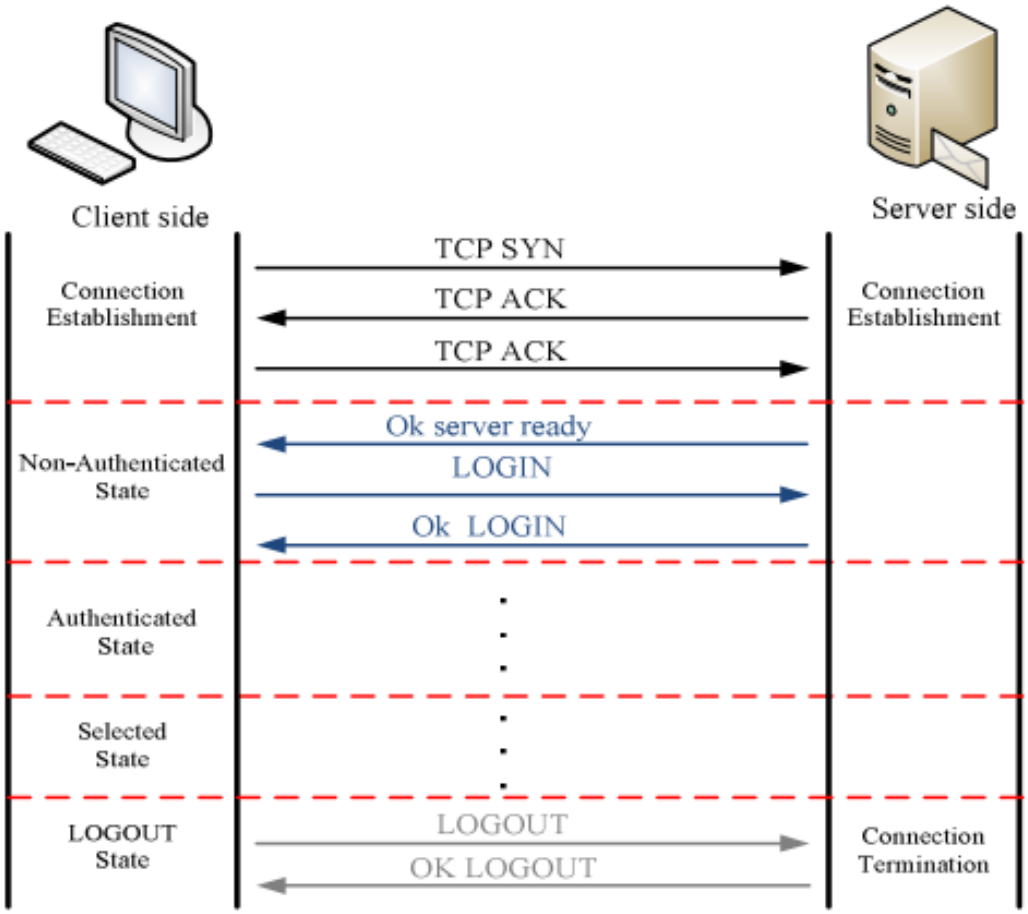
Po navázání TCP spojení klient čeká na uvítací zprávu OK od serveru, která klienta informuje, že emailový server je připraven zahájit příkazový dialog mezi nimi. Po obdržení uvítací zprávy přejde klient do neověřeného stavu. V tomto stavu klient vydá příkazy AUTHENTICATE nebo LOGIN, aby přešel do stavu Authenticated (autentizován). [37]

Příkaz AUTHENTICATE používá klient k zjištění, zda daný server poskytuje mechanismy ověřování. Pokud emailový server ověřování podporuje, odpoví pomocí odpovědi s označením OK, v opačném případě server odpoví pomocí odpovědi NO a mechanismus odmítne. [37]

V tomto stavu je nejběžnější příkaz LOGIN, kdy klient odešle serveru příkaz LOGIN s argumenty uživatelské jméno a heslo. Pokud jsou uživatelské jméno a heslo správné, emailový server odpoví pomocí odpovědi OK a klient přejde do stavu Authenticated, v opačném případě emailový server odpoví pomocí tagu NO a klient musí příkaz odeslat znovu. [37]

Nyní je komunikace ve stavu Authenticated. Klient může zahájit proces dialogu příkazů, dokud klient nedokončí proces načtení emailových zpráv ze serveru. Jakmile již nejsou žádné další emailové zprávy, které by bylo třeba načíst, klient použije příkaz LOGOUT, aby informoval server, že splnil své úkoly, a požádá o ukončení relace. Server

odpoví pomocí neoznačené odpovědi BYE a následně odešle označenou odpověď OK. [13] [37]



Obrázek 3.5: IMAP4 komunikace [13]

3.4.2 Výhody IMAP protokolu

IMAP protokol byl vytvořen, aby eliminoval nedostatky protokolu POP3. Proto ve srovnání s tímto protokolem a vzhledem k používání internetové pošty v dnešní době, se IMAP pyšní převážně výhodami.

- Pošta je uložena na vzdáleném serveru, takže je přístupná z více zařízení.
- Všechny změny jsou sledovány na serveru, což zabraňuje duplicitním schránkám, odeslaným zprávám existujícím pouze na jednom zařízení a podobným problémům.
- Rychlejší přehled, protože se stahují pouze záhlaví, dokud není obsah výslovně požadován.

- Pošta je automaticky zálohována, pokud je server správně spravován.
- Šetří místo v místním úložišti tím, že nevyžaduje, aby počítač stahoval všechny zprávy.
- V případě potřeby je možnost ukládat poštu lokálně.

Kapitola 4

Zabezpečení emailových zpráv

První systémy elektronické pošty byly velmi jednoduché. Obsahovaly pouze základní funkce elektronické pošty bez jakýchkoli bezpečnostních služeb. V dnešní době většina lidí a organizací používá elektronickou poštu pro různé potřeby výměny citlivých informací. [38]

Cílem bezpečnosti komunikace je zajistit, aby nikdo nemohl číst, nebo ještě hůře, tajně měnit zprávy pro ostatní.

V důsledku toho byly stávající emailové systémy vyvinuty tak, aby byly pokročilejší a vyhovovaly potřebám uživatelů. Systémy elektronické pošty se však stávají terčem mnoha hrozeb a odposlechů. Proto se zvýšila potřeba vysoce zabezpečených emailových systémů a aplikací. [39]

4.1 Typy šifrovaní

Šifrování je metoda, která se používá k ochraně důvěrnosti dat. Převádí (neboli šifruje) data a informace z prostého textu do nečitelného formátu — tzv. šifrovaného textu. Ten mohou číst pouze oprávněné osoby, které mají dešifrovací klíč. Jakmile jsou data zašifrována, jsou nečitelná a nerozpoznatelná. Šifrování je vratná transformace. Proces, při kterém jsou nečitelná data (šifrovaný text) transformována na čitelná (prostý text), se nazývá dešifrování. [40]

Šifrování je považováno za účinný proces k zajištění bezpečnosti dat. K přečtení zašifrovaného souboru je zapotřebí tajný klíč nebo šifrovací klíč k dešifrování souboru. Moderní šifrování se provádí pomocí šifrovacího algoritmu a klíče.

Pro šifrování a dešifrování chráněných dat jsou běžné dvě techniky — asymetrické a symetrické šifrování.

4.1.1 Symetrické šifrování

Při šifrování se symetrickým klíčem má každý uživatel používající šifrovací systém kopii jediného tajného klíče. Ten se používá k šifrování i dešifrování dat. Výhoda symetrického šifrování oproti asymetrickému je zejména v rychlosti. [40]

Vzhledem k tomu, že tajný klíč je sdílen s odesílatelem i příjemcem, stává se jeho použití riskantním. Při symetrickém šifrování se doporučuje používat klíče o délce 128 bitů nebo delší, aby se předešlo případným pokusům o prolomení. Mezi nejběžnější šifrovací algoritmy patří AES, RC4, DES, 3DES, RC5 a RC6. Z těchto algoritmů je AES (Advanced Encryption Standard) pokládáno za nejbezpečnější. Symetrické šifrování se používá v případech, kdy je rychlost procesu požadována více než bezpečnost. Mezi nejčastější použití symetrického šifrování patří např: [40]

- Bankovní a finanční organizace: Šifrování informací o kreditních kartách a osobně identifikovatelných údajů (PII) k dokončení transakcí.
- Ukládání dat: šifrování dat v klidovém stavu.

4.1.2 Asymetrické šifrování

Je také známá jako kryptografie s veřejným klíčem. Používá dva odlišné šifrovací klíče, které spolu souvisejí. První klíč je znám jako veřejný klíč a druhý klíč je znám jako soukromý klíč.

Veřejný klíč je k dispozici každému, kdo chce odesílateli poslat zprávu. Druhý soukromý klíč zůstává tajný, zná ho pouze odesílatel. Zprávu zašifrovanou veřejným klíčem lze dešifrovat pomocí soukromého klíče. Zprávu zašifrovanou soukromým klíčem lze dešifrovat pomocí veřejného klíče. Bezpečnost veřejného klíče není vyžadována. Proto může být k dispozici ke stažení na nějakém veřejném serveru a může volně procházet internetem. [40]

Asymetrické šifrování je považováno za nejlepší volbu pro přenos informací. Používá se při komunikaci klienta a serveru. K vyhledání serveru je vytvořen certifikát, který obsahuje profil a informace o organizaci. Po síti se odešle dotaz druhé straně, která jí pošle zpět certifikát.

Protokol TLS používá asymetrické i symetrické šifrování a to zejména z důvodu rychlosti. Velkou výhodou asymetrického šifrování je navázat šifrované spojení s uživatelem, kterého druhý uživatel nikdy neviděl a nemá tedy jiný bezpečný komunikační kanál, kterým by předal symetrický klíč.

Mezi algoritmy používající tuto techniku patří RSA, který se používá pro šifrování a ověřování, a Pretty Good Privacy (PGP), který se používá např. k zabezpečení emailů. [40]

Asymetrické šifrování se používá v případech, kdy má bezpečnost přednost před rychlostí, mezi nejčastější použití asymetrického šifrování patří např:

- Digitální podpis: používá se k potvrzení identity podpisu.
- Blockchain: potvrzení identity pro autorizační transakce u kryptoměn.
- Infrastruktura veřejných klíčů: autorizace šifrovacích klíčů prostřednictvím digitálních certifikátů.

4.2 SSL a TLS

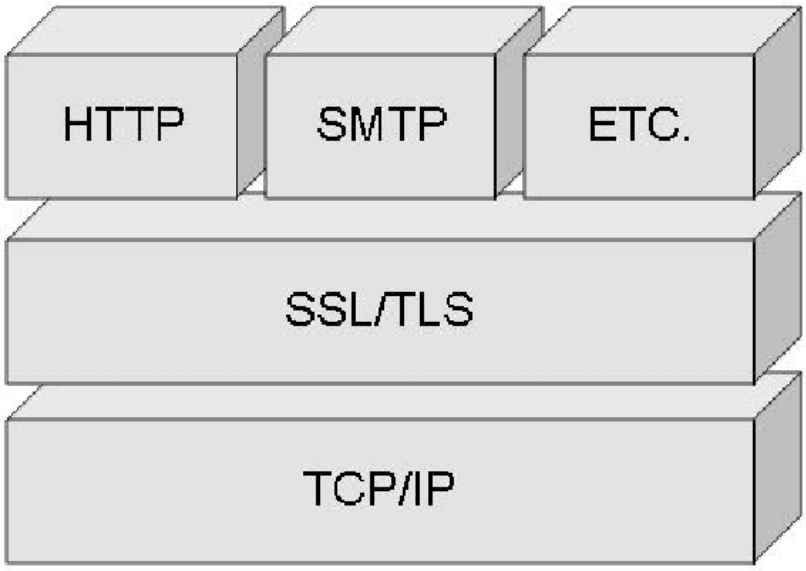
Protokoly SSL a TLS zajišťují šifrování komunikace a autentizaci klient-server. SSL bylo poprvé představeno společností Netscape Communications v roce 1994 a bylo dvakrát revidováno. V roce 1996 Internet Engineering Task Force (IETF) ustavila pracovní skupinu TLS, aby formalizovala a posunula protokol SSL na úroveň internetového standardu. [41]

Protokol TLS verze 1.0 je formálně specifikován v normě RFC 2246 [42], která byla publikován v roce 1999. Je založen z velké části na SSL verzi 3.0. Jak se píše i v dokumentaci — SSL verze 3 a TLS verze 1 jsou v podstatě totožné. [42]

Většina dnešní komunikace založená na TCP/IP paketu používá k zajištění bezpečnosti nějakou z verzí TLS protokolu. Aktuální verze protokolu TLS je vezrze 1.3, ale nejpoužívanější verze je stále verze 1.2. [43]

Připojení SSL/TLS pracuje na 4. až 7. vrstvě ISO/OSI modelu a vytváří spojení mezi klienty a servery. Každá SSL/TLS relace je spojena s jedním SSL/TLS připojením. SSL/TLS handshake protokol se používá k vytvoření relace výměnou několika parametrů.

Každá relace je udržována převážně dvěma stavy. Stav relace se zabývá řadou parametrů, jako je identifikátor relace, kompresní techniky, specifikace šifry atd. Parametr stavu připojení zahrnuje tajné klíče MAC pro odesílání serveru a klienta, inicializační vektory, sekvenční čísla atd.[41]



Obrázek 4.1: Zařazení SSL a TLS v TCP/IP modelu [41]

4.3 TLS

TLS je vylepšená verze SSL. Má stejnou architekturu a pod-protokoly s výjimkou některých změn např. v bezpečnostních parametrech a výpočtu MAC (message authentication code — kontrolní součet).

Protokol TLS se skládá ze dvou skupin dílčích protokolů:

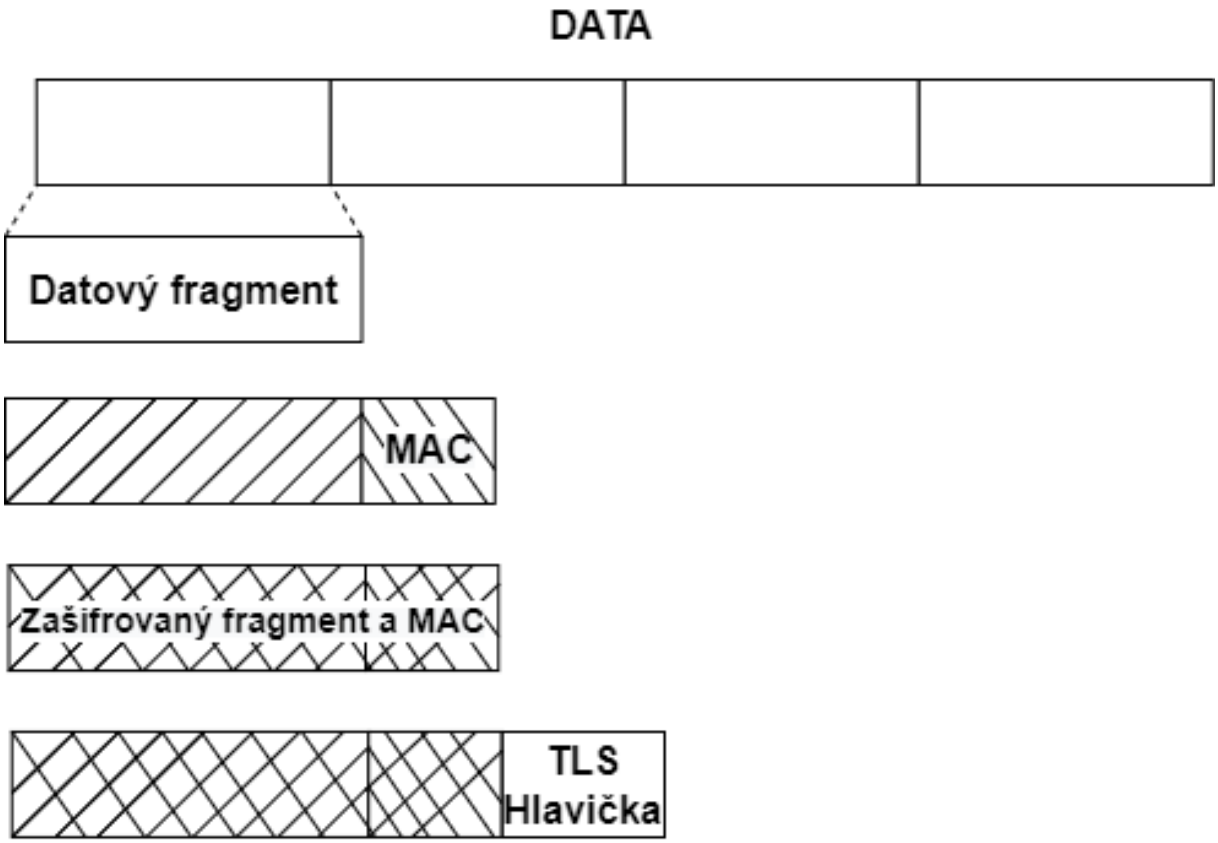
- TLS Record protocol
- Řídící protokoly TLS

4.3.1 TLS Record protocol

TLS Record protocol poskytuje standardy pro formát TLS zprávy. Funguje jako základ pro řídicí protokoly a poskytuje důvěrnost a integritu zprávám vyšší vrstvy.[42] [41]

Důvěrnost dat se zajišťuje pomocí šifrování symetrickým klíčem a integrita dat pomocí klíčového kontrolního součtu (MAC). Klíče jsou generovány pro každou relaci na základě bezpečnostních parametrů dohodnutých během TLS handshake procedury. [41]

Record protokol se také používá k zapouzdření různých protokolů vyšší vrstvy – zejména protokolu TLS Handshake – v tomto případě jej lze použít bez šifrování nebo ověřování zpráv. Další protokoly zapouzdřené v Record Protocol jsou Alert Protocol a Change Cipher Spec Protocol. [44]



Obrázek 4.2: TLS Record protocol

Struktura tvorby SSL/TLS paketů je rozdělena do pěti částí. [44]

- Data se rozdělí do několika částí (tzv. fragmentů).
- Jednotlivé části se zkomprimují (Ve výchozím nastavení je komprese v SSLv3.0 a všech verzích TLS zakázána)
- Vypočítá se MAC.
- Dohromady se zašifrují obě části společně s odpovídajícím MAC.
- Připojí se SSL hlavička.

V místě příjemce jsou tyto procesy prováděny v opačném směru před doručením původních zpráv příjemci. [44]

MAC (HMAC)

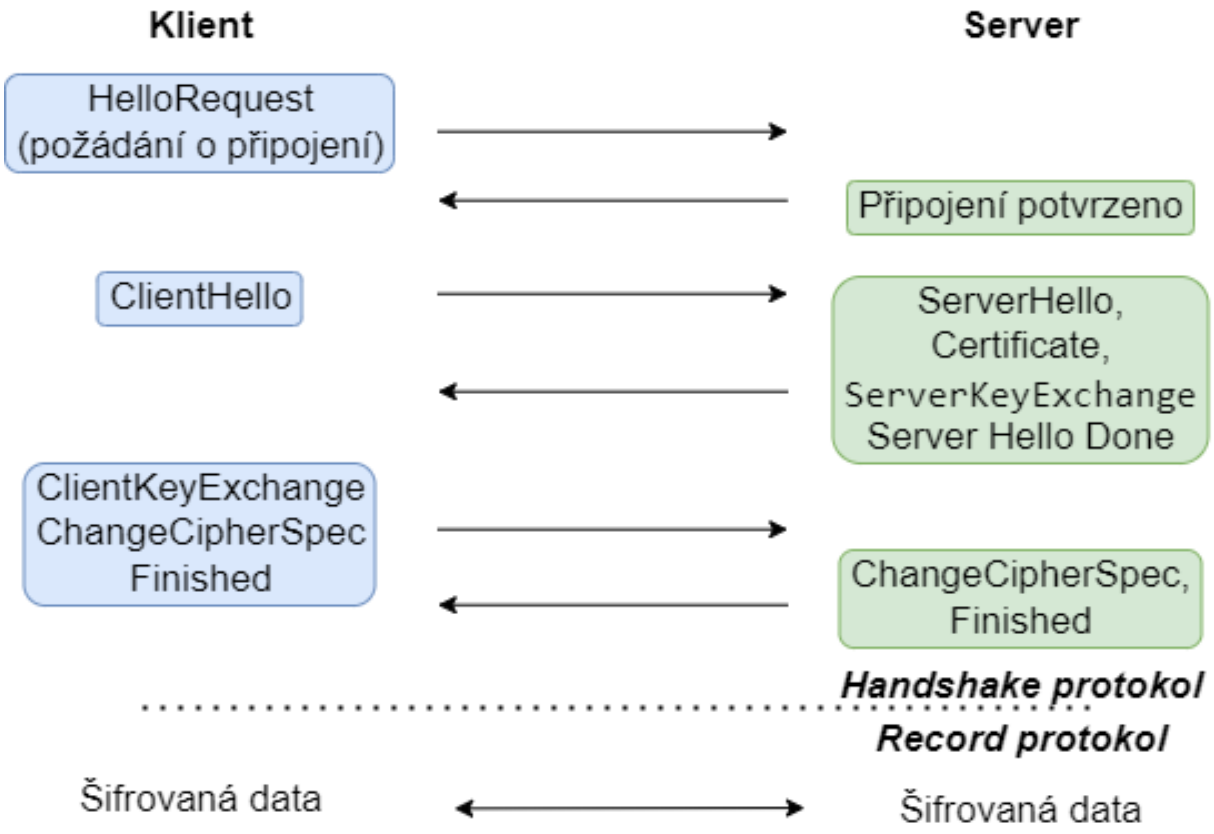
Message authentication code (MAC) je metoda, která se používá ke kontrole pravosti a integrity zprávy. Přijímá dva vstupní parametry: tajný klíč a zprávu libovolné délky. Výsledek se nazývá tag. Příjemce má také tajný klíč a může jej použít ke zjištění jakýchkoli změn v obsahu zprávy. MAC se někdy nazývá kontrolní součet, kryptografický kontrolní součet nebo chráněný kontrolní součet. [45]

Pokud se MAC tag odesílatele a vypočtený MAC tag příjemce shodují, nikdo se zprávou nemanipuloval. Pokud se neshodují, někdo zprávu během přenosu pozměnil.

4.3.2 TLS Handshake

Nejdůležitější část navázání zabezpečeného spojení se nazývá handshake — podání ruky. Během této procedury si server a klient vyměňují důležité informace sloužící k určení vlastností spojení. Uvedený příklad vychází z handshakeu webového prohlížeče s http(s) serverem, ale totéž platí pro všechny ostatní SSL/TLS handshake procedury.

Pro účely vysvětlení procesu bylo vybráno spojení TLS 1.2, nikoli nejnovější protokol TLS 1.3. Proces používaný v protokolu TLS 1.2 je téměř stejný u všech předchozích verzí protokolu SSL/TLS. V nejnovější verzi 1.3 byl však výrazně zjednodušen. Jak TLS 1.2 handshake tak i 1.3 jsou z hlediska bezpečnosti prakticky na stejné úrovni. Dané zjednodušení přineslo hlavně zrychlení celé procedury. [46]



Obrázek 4.3: TLS handshake schéma

ClientHello

TLS handshake začíná tím, že klient odešle na server zprávu ClientHello. Tato zpráva obsahuje následující pole: [46] [47]

- Verze TLS (client_version): Klient odešle seznam všech verzí protokolu TLS, které podporuje, přičemž preferovaná verze je na seznamu na prvním místě.
- Random (random): Jedná se o 32bajtové náhodné číslo. Náhodné číslo klienta a náhodné číslo serveru se později použijí k vygenerování klíče pro šifrování.
- ID relace (session_id): Toto je ID relace, které se použije pro připojení. Pokud není session_id prázdné, server vyhledá dříve uloženou relaci v mezipaměti a pokud najde shodu, obnoví danou relaci.
- Metody komprese (compression_methods): Jedná se o metodu, která bude použita pro kompresi paketů. Z důvodů možných útoků právě skrz komprimaci se už příliš nepoužívá.

- Sady šifer (cipher_suites): Sady šifer jsou kombinace kryptografických algoritmů. Každá sada šifer obvykle obsahuje jeden kryptografický algoritmus pro každou z následujících úloh: výměnu klíčů, ověřování, hromadné šifrování dat a ověřování zpráv. Klient odešle seznam všech sad šifer, které podporuje, v pořadí podle jejich preference. To znamená, že klient by v ideálním případě preferoval navázání spojení pomocí první odeslané sady šifer.

  Sady šifer jsou označeny řetězci. Vzorový řetězec sady šifer je následující:
  TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.
  Tento řetězec obsahuje následující informace:

  - TLS je používaný protokol
  - ECDHE je algoritmus výměny klíčů (Eliptic Curve Diffie-Hellman — Diffie-Hellmanova eliptická křivka).
  - ECDSA je ověřovací algoritmus (Elliptic Curve Digital Signature Algorithm — algoritmus digitálního podpisu).
  - AES_128_GCM je algoritmus šifrování dat (Advanced Encryption Standard 128 bit Galois/Counter Mode).
  - SHA256 je algoritmus MAC (Message Authentication Code) (Secure Hash Algorithm 256 bit).

- Rozšíření(extensions): Klient si může vyžádat další funkce pro připojení. To lze provést prostřednictvím rozšíření, jako jsou podporované skupiny pro kryptografii eliptických křivek, formáty bodů pro kryptografii eliptických křivek, podpisové algoritmy a další. Pokud server nemůže poskytnout dodatečnou funkci, může klient v případě potřeby přerušit handshake. [46]

ServerHello

Server odešle tuto zprávu jako odpověď na zprávu ClientHello, pokud se mu podaří najít přijatelnou sadu algoritmů. Pokud takovou shodu nenajde, odpoví zprávou o selhání handshake. [47]

- Verze serveru (server_version): Server vybere preferovanou verzi protokolu TLS z verzí předložených klientem.

- Random (random): Jedná se o 32bajtové náhodné číslo. Náhodné číslo serveru a náhodné číslo klienta se později použijí k vygenerování šifrovacího klíče. V původní specifikaci TLS 1.2 měly první 4 bajty představovat aktuální datum a čas klienta

a zbývajících 28 bajtů mělo být náhodně vygenerované číslo. IETF to však později nedoporučila.

- ID relace(session_id): Pokud ID relace klienta nebylo prázdné, server vyhledá dříve uložené relace v mezipaměti, a pokud je nalezena shoda, použije se toto ID k obnovení relace. Pokud bylo ID relace klienta prázdné, může server vytvořit novou relaci a odeslat ji v ID relace serveru.
- Sady šifer(cipher_suite): Server vybere sadu šifer ze sad šifer zaslaných v ClientHello.
- Metody komprese (compression_method): Server vybere metodu komprese z metod komprese zaslaných v ClientHello.
- Rozšíření (extensions): V seznamu serveru se mohou objevit pouze rozšíření nabízená klientem. nabízená klientem. [46] [47]

Certifikát serveru

Server nyní odešle klientovi podepsaný certifikát TLS/SSL, který prokazuje jeho identitu. Obsahuje také veřejný klíč serveru. [46] [47]

Certifikát klienta (nepovinné)

Ve výjimečných případech může server vyžadovat ověření klienta pomocí klientského certifikátu. V takovém případě klient poskytne serveru svůj podepsaný certifikát. [46] [47]

Výměna klíče serveru

Zpráva o výměně klíče serveru je odeslána pouze v případě, že certifikát poskytnutý serverem není dostatečný pro výměnu předřazeného tajemství klienta. [46] [47]

ServerHello Done

Server odešle tuto zprávu klientovi, aby potvrdil, že zpráva Server Hello je hotová. [46] [47]

Výměna klientských klíčů

Zpráva ClientKeyExchange je odeslána ihned po přijetí zprávy Server Hello Done ze serveru. Pokud si server vyžádá klientský certifikát, je zpráva Client Key Exchange odeslána až poté. Během této fáze klient vytvoří předřazený klíč. [46]

- Pre-Master Secret: Tento klíč (pre-master secret) vytváří klient (způsob vytvoření závisí na sadě šifer) a poté jej sdílí se serverem. [47]

  Před odesláním předřazeného tajemství serveru jej klient zašifruje pomocí veřejného klíče serveru získaného z certifikátu poskytnutého serverem. To znamená, že zprávu může dešifrovat pouze server, protože pro výměnu předřazeného tajemství se používá asymetrické šifrování. [46] [47]

- Hlavní tajemství (Master Secret): Poté, co server obdrží klíč pre-master secret, použije k dešifrování svůj soukromý klíč. Nyní klient a server vypočítají hlavní tajný klíč na základě dříve vyměněných náhodných hodnot (Client Random a Server Random) pomocí pseudonáhodné funkce (PRF). PRF je funkce, která slouží ke generování libovolného množství pseudonáhodných dat. [47]

  Hlavní tajný klíč o délce 48 bajtů pak klient i server použijí k symetrickému šifrování dat po zbytek komunikace. [46]

Klient a server vytvoří sadu 3 klíčů:

- client_write_MAC_key: Ověřování a kontrola integrity
- server_write_MAC_key: Autentizace a kontrola integrity
- client_write_key: Šifrování zprávy pomocí symetrického klíče
- server_write_key: Šifrování zprávy pomocí symetrického klíče
- client_write_IV: Inicializační vektor používaný některými šiframi.
- server_write_IV: Inicializační vektor používaný některými šiframi.

Klient i server použijí hlavní tajemství pro generování klíčů relací, které budou sloužit k šifrování/dešifrování dat. [47]

Změna šifry klienta

V tomto okamžiku je klient připraven přejít do zabezpečeného šifrovaného prostředí. Ke změně šifrování se používá protokol ChangeCipherSpec. Veškerá data odesílaná klientem budou od této chvíle šifrována pomocí symetrického sdíleného klíče. [46] [47]

Klient Handshake dokončen

Poslední zpráva procesu handshake od klienta znamená, že handshake je dokončen. Je to zároveň první šifrovaná zpráva zabezpečeného spojení. [46] [47]

Změna šifrovací specifikace serveru

Server je rovněž připraven přejít do šifrovaného prostředí. Veškerá data odesílaná serverem budou od této chvíle šifrována pomocí symetrického sdíleného klíče. [46] [47]

Server Handshake dokončen

Poslední zpráva procesu handshake ze serveru (odeslaná šifrovaně) znamená, že handshake je dokončen. [46] [47]

4.4 OpenPGP

Pretty Good Privacy (PGP) je šifrovací systém používaný k odesílání šifrovaných emailů i k šifrování citlivých souborů. Od svého vynálezu v roce 1991 se PGP stal v podstatě standardem pro zabezpečení emailů.

Od roku 1997 byla v rámci standardizační organizace Internet Engineering Task Force (IETF) vytvořena pracovní skupina OpenPGP. Ta definovala tento standard, který byl dříve soukromým produktem. V uplynulém desetiletí se PGP a později OpenPGP stal standardem pro téměř všechny podepsané nebo zašifrované emaily na světě. [48]

Popularita PGP je založena na dvou faktorech. Prvním je, že systém byl původně k dispozici jako freeware, a tak se rychle rozšířil mezi uživateli, kteří chtěli další úroveň zabezpečení svých emailových zpráv. Druhým důvodem je, že vzhledem k tomu, že PGP nabízí i šifrování veřejným klíčem, umožňuje uživatelům, kteří se nikdy nesetkali, posílat si navzájem šifrované zprávy, aniž by si museli vyměňovat své soukromé šifrovací klíče. [49]

OpenPGP také definuje standardní formát certifikátů, který na rozdíl od většiny ostatních formátů certifikátů umožňuje vytvářet tzv. sítě důvěry. Formáty a použití OpenPGP jsou specifikovány v mnoha RFC a návrzích IETF1 , takže tyto standardy může implementovat jakákoli společnost, aniž by musela komukoli platit licenční poplatky. [48]

4.4.1 Princip OpenPGP

Protokol PGP má některé funkce společné s jinými šifrovacími systémy jako například se šifrováním Kerberos (které se používá k ověřování uživatelů sítě) a šifrováním SSL/TLS (které se používá k zabezpečení webových stránek). [50]

Na základní úrovni používá šifrování PGP kombinaci dvou forem šifrování: šifrování symetrickým klíčem a šifrování veřejným klíčem.

Na nejvyšší úrovni funguje šifrování PGP takto:

- Nejprve PGP vygeneruje náhodný klíč relace pomocí jednoho ze dvou (hlavních) algoritmů. Tento klíč je obrovské číslo, které nelze uhodnout, a používá se pouze jednou.

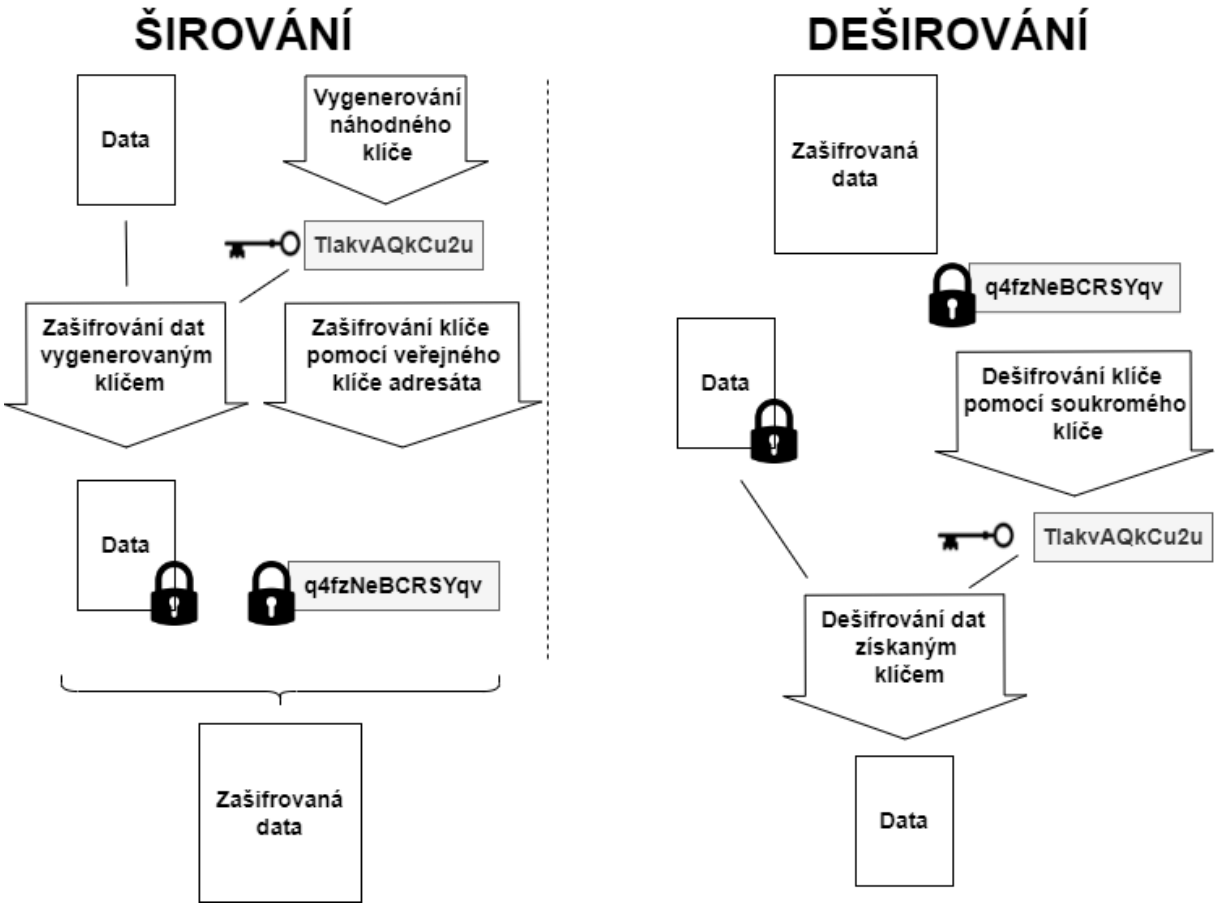
- Poté je tento klíč relace zašifrován. K tomu se použije veřejný klíč zamýšleného příjemce zprávy. Veřejný klíč je vázán na identitu konkrétní osoby a kdokoli jej může použít k odeslání zprávy.

- Odesílatel pošle svůj zašifrovaný klíč relace PGP příjemci a ten jej může dešifrovat pomocí svého soukromého klíče. Pomocí tohoto klíče relace je nyní příjemce schopen dešifrovat skutečnou zprávu.

Důvodem použití kombinace veřejného a soukromého klíče má své opodstatnění zejména z hlediska rychlosti. Kryptografie s veřejným klíčem je pomalejší než symetrické šifrování. Použití symetrického šifrování však vyžaduje, aby odesílatel sdílel šifrovací klíč s příjemcem v prostém textu, a to by bylo nezabezpečené. Šifrováním symetrického klíče pomocí (asymetrického) systému veřejného klíče tedy PGP kombinuje rychlost symetrického šifrování s bezpečností kryptografie veřejného klíče.

Původní program PGP byl nabízen ve dvou verzích, jedna používala pro výměnu klíčů algoritmus Rivest-Shamir-Adleman (RSA) a druhá algoritmus Diffie-Hellman. Za verzi RSA musel PGP platit společnosti RSA licenční poplatek. [50]

Verze PGP RSA používá ke generování hash kódu algoritmus MD5. Verze PGP Diffie-Hellman používala ke generování hashovacího kódu algoritmus SHA-1; ani jeden z těchto hashovacích algoritmů není dnes považován za bezpečný. [50] [49] [51]



Obrázek 4.4: OpenPGP schéma šifrování

4.4.2 Ověřování digitálního podpisu

Souvisejícím využitím protokolu PGP je jeho použití pro ověřování emailů. Pokud si uživatel není jistý totožností osoby, která mu posílá zprávu, může k jejímu ověření použít vedle PGP také digitální podpis.

Digitální podpisy fungují tak, že pomocí algoritmu zkombinují soukromý klíč odesílatele s daty, které odesílá. Nejdříve se vytvoří hash z odesílané zprávy. Ten je pak zašifrován pomocí soukromého klíče odesílatele. [51]

Příjemce zprávy pak může tato data dešifrovat pomocí veřejného klíče odesílatele. Pokud byl při přenosu změněn byť jen jediný znak zprávy, příjemce to pozná. To může znamenat, že odesílatel není tím, za koho se vydává, že se pokusil zfalšovat digitální podpis nebo že zpráva byla zfalšována.

4.5 Existující řešení šifrovaní emailových zpráv

Co se týče popularity a snadnosti použití, nelze překonat hlavní poskytovatele emailových služeb, jako jsou Google a Microsoft.

Tyto služby však není možné označit jako bezpečné. Komunikace vedená prostřednictvím elektronické pošty není obvykle zabezpečena end-to-end šifrováním. Existuje ale řada dalších problémů týkajících se soukromí a bezpečnosti u nejznámějších poskytovatelů. Například služba Gmail od společnosti Google umožňuje poskytovatelům služeb třetích stran skenovat vaše soukromé emaily a zobrazovat tak více personalizované reklamy. [52]

Existuje však několik poskytovatelů emailových služeb, kteří nabízejí vyšší zabezpečení.

4.5.1 PreVeil

Systém PreVeil je postaven na asymetrickém šifrováním. Každý uživatel má soukromý a veřejný klíč - veřejný klíč je zveřejněn všem ostatním uživatelům PreVeilu. Soukromý klíč je uchováván na počítači uživatele. Chce-li uživatel někomu poslat email, zašifruje danou zprávu veřejným klíčem adresáta. Tím se zajistí, že jej pak bude moci dešifrovat pouze adresát pomocí svého tajného klíče. K emailům můžou uživatelé přistupovat z více zařízeních, ale musí zajistit bezpečný přenos svého soukromého klíče. [53]

Průběh zaslání emailu

Vysvětlený průběh komunikace, je popsán na dvou imaginárních uživatelích — Alice a Bob. Pokud chce Alice poslat Bobovi email, vygeneruje nejprve náhodný symetrický šifrovací klíč K. Alice zašifruje M pod K, čímž vznikne C_M, a poté zašifruje K pod veřejným klíčem Boba PK_B, čímž vznikne C_K. Nakonec Alice vytvoří podpis σ C_M pomocí jejich tajného klíče SK_A. Alice pak odešle (C_M, CK, σ) serveru PreVeil.

Pro získání Alicina emailu si Bob stáhne (C_M, CK, σ) ze serveru a provede výše uvedený proces opačně. Nejprve získá Alicin veřejný klíč a použije jej k ověření podpisu σ. Svůj soukromý klíč použije k dešifrování C_K a získá tak K, který pak použije k dešifrování CM a získání M. [53]

4.5.2 SecureMyEmail

SecureMyEmail nabízí řešení, které se liší od ostatních poskytovatelů. Umožňuje totiž vytvořit zabezpečení a šifrovaní emailu až pro pět již existujících emailových účtů. Uživatel si tedy nemusí měnit adresu, aby mohl posílat zabezpečené emaily.

Služba poskytuje účtům další vrstvu zabezpečení: šifrování s nulovou znalostí. To znamená, že obsah emailů nemůže vidět nikdo jiný než určený příjemce. Jsou zašifrovány u zdroje a dešifrovat je lze, až když dorazí k cíli. [54]

Celá architektura bezpečnosti je postavena na OpenPGP standardu. Při vytváření účtu na www.securemyemail.com je uživateli vygenerován veřejný a soukromý klíč. Tento pár klíčů je pak používán dle pravidel asymetrického šifrování k zašifrování a podepisování elektronických zpráv. [55]

4.5.3 ProtonMail

Jedná se nejspíše o nejznámějšího poskytovatele bezpečné elektronické pošty. Společnost ProtonMail byla založena v roce 2014 v Evropské organizaci pro jaderný výzkum (CERN), je bezpečný poskytovatel emailu s end-to-end šifrováním a zárukou nulového přístupu. Služba vznikla v reakci na úniky informací od Edwarda Snowdena. Její datová centra sídlí ve Švýcarsku v podzemním bunkru dostatečně silném na to, aby přežil i jaderný útok. [56]

Kromě silného end-to-end šifrování s nulovou znalostí má ProtonMail mnoho užitečných funkcí. Uživatel může např. nastavit automatický odesílač, vytvořit vlastní filtr spamu a mnoho dalších. Jedním z charakteristických vlastností služby ProtonMail jsou tzv. samodestrukční emaily. Jedná se o zprávy, které jsou po uplynutí stanovené doby automaticky odstraněny ze schránky příjemce. [57]

ProtonMail nabízí aplikaci Bridge. Tuto aplikace může uživatel použít k šifrování všech emailů při jejich příchodu a odchodu z počítače. Po instalaci a nastavení aplikace může uživatel používat svého emailového klienta, například Thunderbird s jistotou, že emaily jsou při odesílání přes internet plně šifrovány. [57]

Kapitola 5

Proxy servery

Proxy server je obvykle kombinace hardwaru a softwarových aplikací. Slouží jako prostředník v síťové komunikaci mezi klientem a serverem. Slovo proxy znamená „jednat jménem jiného“. Proxy server jedná z pravidla jménem uživatele. Všechny požadavky na jím požadovaný server jdou nejprve na proxy server, který požadavek vyhodnotí a předá dál. Stejně tak se odpovědi vracejí zpět na proxy server a poté k uživateli.

Základním principem činnosti proxy serveru je přijímání požadavků klienta. Vůči klientovi se proxy tváří jako server. Tyto požadavky analyzuje a následně odesílá do uživatelem požadované destinace, vůči které se proxy chová jako klient. Odpovědi ze serveru pak předává původnímu klientovi a to v původní nebo upravené podobě.

Proxy server většinou pracuje na aplikační vrstvě ISO/OSI modelu a analyzuje příchozí požadavky. Takovýto server se označuje jako aplikační proxy server a pracuje se stejným aplikačním protokolem jako obsluhovaní klienti. Nevýhodou tohoto typu proxy serveru je jeho omezené využití. [58]

5.1 SOCKS5

Kromě aplikačních proxy serverů existují i servery nezávislé na aplikační vrstvě. Ty poskytují pouze transportní pakety bez znalosti protokolů aplikační vrstvy. Jejich použití však vyžaduje použití specifického komunikačního protokolu pro komunikaci s proxy serverem.

Typickým představitelem univerzálního proxy protokolu je protokol SOCKS. Tento standard pracuje na 5. vrstvě ISO/OSI modelu. SOCKS směruje síťové pakety mezi klientem a serverem prostřednictvím proxy serveru. [58]

Jedno z využití tohoto typu proxy serveru je jako alternativa k VPN. Jelikož přeposílá datové pakety místo skutečného klienta, tak zajišťuje, že skutečná IP adresa klienta zůstane skryta a k internetu se přistupuje pod IP adresou proxy serveru. [59]

5.2 Emailové proxy servery

Emailový proxy server je speciálním typem agenta pro přenos pošty (MTA). Takovýto proxy server podobně jako jiné typy proxy serverů předávají relace jiným MTA. SMTP proxy servery, na rozdíl od plnohodnotných SMTP serverů, nepoužívají metodu ukládání a předávání.

Když proxy server zahájí spojení s klientem, zároveň zahájí další SMTP spojení s cílovým serverem. Veškeré chyby nebo stavové informace z cílového serveru budou prostřednictvím proxy serveru předány zpět odesílajícímu uživateli. [60]

Existují dva základní typy SMTP proxy serverů - proxy pro odchozí poštu a proxy pro příchozí poštu.

5.2.1 SMTP proxy pro příchozí poštu

Emailové proxy servery často slouží jako počáteční síťová vrstva emailového systému, která zpracovává spojení od klientů před předáním dat druhé vrstvě poštovních serverů. Proxy servery často představují první a často i jedinou obrannou vrstvu v systému filtrování příchozího spamu.

Mohou analyzovat zprávy pomocí filtru obsahu spamu nebo antivirového programu, blokovat nebo omezovat rychlost spojení pomocí DNS black listů a reputačních systémů. Dále také mohou fungovat jako vyrovnávači zátěže (load balancer) spojení, což zabraňuje přetížení poštovních serverů. [61]

SMTP Tarpitting

Některé proxy servery implementují správu TCP spojení — tzv. řízení toku. Řízení TCP spojení v kontextu protokolu SMTP obvykle zahrnuje omezování šířky pásma a zavádění zpoždění v odpovědích na SMTP příkazy. Tato metoda je známá jako tarpitting.

Jedná se o proces zpomalení hromadného doručování emailů. V rámci ochrany proti spamu se poštovní server rozhodne chvíli počkat, než email doručí do schránky příjemce. Tato prodleva se nazývá tarpit a její trvání se označuje jako tarpit time.

Tarpit time se u jednotlivých serverů liší. Některé starší servery reagují pomalu jednoduše proto, že nejsou přizpůsobeny modernějšímu používání emailu. Ve většině případů však server odpoví do pěti sekund. I tak by ale pět sekund prodlevy na jednu adresu může představovat významnou překážku pro spammery, jejichž záměrem je odeslat co nejvíce emailů za co nejkratší dobu. Pokud například spammer odešle stejný email 20 000 příjemcům, při pěti sekundách na příjemce by celý proces mohl trvat téměř 30 hodin. Spamovací systémy (tzv. spamboti) jsou stavěny na rychlost a některé z nich se dokonce vzdávají odesílání hromadných emailů, pokud narazí na výrazné zpoždění. [62]

To může obtěžovat uživatele rozesílající legitimní emaily. SMTP tarpitting funguje jak na platné, tak na neplatné emailové adresy, takže po všem tom čekání můžete nakonec obdržet zprávu o nedoručení (Non-Delivery Report, NDR), známou také jako bounce email.

Síťový tarpitting může být náročný na implementaci v rámci emailového serveru, protože každé SMTP spojení je zpracováváno pomaleji než obvykle, což často zdržuje cenné systémové zdroje, jako je paměť a procesor. Vzhledem k tomu, že SMTP proxy servery lze implementovat pomocí lehčích programovacích technik, lze obsluhovat mnohem více spojení se stejnými prostředky jako s mnohem menším počtem spojení v kontextu plnohodnotného emailového serveru. [62]

5.2.2 SMTP proxy pro odchozí poštu

Ke kontrole odchozí pošty se používá SMTP proxy pro odchozí poštu. Filtrování odchozího spamu lze použít k zajištění toho, aby všechny emaily odeslané prostřednictvím skutečného SMTP serveru byly již ověřeny proxy serverem. Proto jsou všechny nevyžádané emaily ověřeny dříve, než se dostanou na server příjemce.

Význam proxy serveru pro odchozí poštu spočívá v tom, že umožňuje organizacím zachovat důvěryhodnost jejich poštovních serverů. Jakýkoli email odeslaný ze škodlivě infikovaného počítače může dále šířit infekci do počítače příjemce. Služba této proxy je využívá jako ochrana před zařazením na DNS černou listinu. [63]

Existují dva typy filtrování odchozích zpráv.

Open source

Existují open-source proxy pro filtrování odchozího spamu, které mohou organizace používat se svým skutečným serverem SMTP. Příkladem takové služby je např. MailCleaner & Proxmox. [63]

Office 365 spam filtr

Nástroj Office 365 od společnosti Microsoft umožňuje každému uživateli využívat filtr odchozího spamu ke kontrole a ověřování všech emailů odeslaných prostřednictvím jeho serveru. Outbound spam office 365 je jednoduchý nástroj, který lze integrovat s vlastním poštovním serverem uživatele pro filtrování příchozího nebo odchozího spamu. [63]

Kapitola 6

Řešení šifrované emailové komunikace

Cílem praktické části je vytvoření proxy serveru, který umí šifrovat odchozí elektronickou poštu. Tento proxy server je rozšíření pro již existující službu GDPRCloud.

6.1 GDPRCloud

Jedná se, jak už název napovídá, o cloudovou službu. Z obecného hlediska se pod pojmem cloud označují servery, které jsou přístupné skrz internet. Z pravidla se jedná o nějaký balíček služeb, které využívají právě výhody, že nejsou umístěny fyzicky na pevném disku uživatele, ale vyskytují se na serverech poskytovatele. Uživatel má pak k těmto službám přístup odkudkoliv, kde je internetové připojení.

GDPRCloud navíc splňuje ještě další důležitou podmínku — je certifikovaný z hlediska kybernetické bezpečnosti. To znamená, že veškerá spojení mezi počítači jsou zabezpečena. [64]

Veškerá hesla zná pouze uživatel popř. lidé, kterým uživatel sdělí své přístupové údaje. Nikdo jiný (tedy ani společnost GDPRCloud, ani internetoví operátoři) je neznají. To zabezpečuje ochranu před přečtením nebo pozměněním jakýchkoliv informací.

V názvu se vyskytuje GDPR z toho důvodu, že myšlenka na vytvoření této softwarové služby vznikla na popud této evropské regulace. GDPR je v podstatě vymezením zodpovědnosti při práci s osobními daty uživatelů. Ve většině případů je tato zodpovědnost směřována na firmy. Jinými slovy se dá říct, že stanovuje právo občana EU předpokládat, že jeho údaje budou bezpečně spravovány a chráněny.

Právě díky důrazu na bezpečnost, má GDPRCloud ambici poskytovat IT služby firmám právě v souladu s pravidly GDPR.

6.1.1 GDPRCloud email

Jedna ze služeb, co GDPRCloud nabízí, je elektronická pošta. Právě email je z dnešního hlediska bezpečnosti jedním z nejzranitelnějších oblastí v IT.

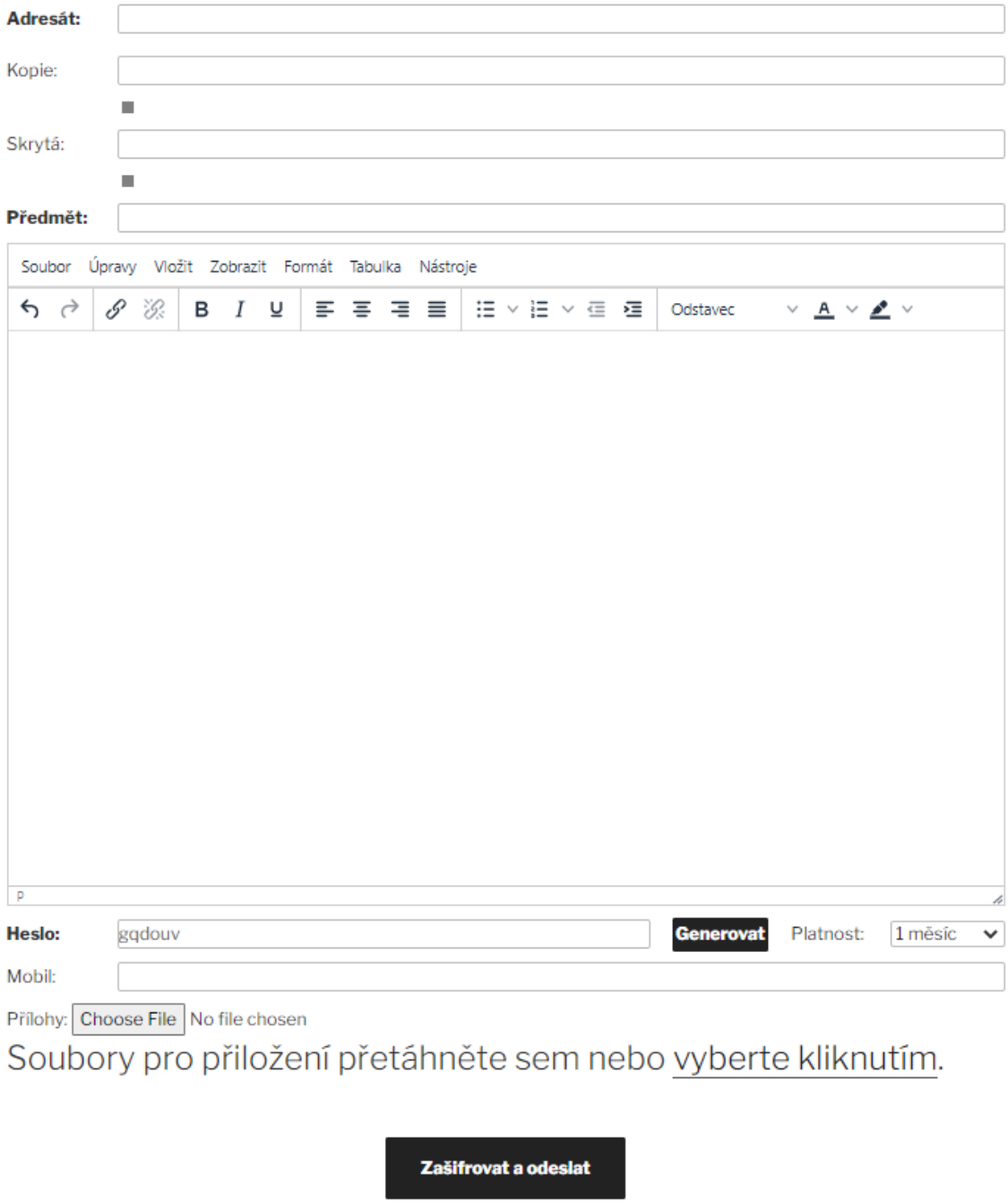
Pokud tedy firma podnikající na půdě Evropské unie má spravovat nějaká data o zákaznících, měla by se vyvarovat sdílením těchto dat přes elektronickou poštu. Ačkoliv GDPR výslovně nevyžaduje šifrování emailu, neexistuje prakticky moc jiných možností než end-to-end šifrování, jak zaručit bezpečnost přenosu dat až k adresátovi. [65]

Jak už bylo zmíněno v úvodu, problémem end-to-end šifrování elektronické pošty je zajištění výměny klíčů. Navíc cílový uživatel ani nemusí end-to-end šifrování podporovat. Z těchto důvodů GDPRCloud nabízí vlastní systém zajištění end-to-end šifrování.

Pro posílání zpráv se používá symetrické šifrování. To znamená že klíč použitý k zašifrování emailové zprávy je potřeba nějakým způsobem doručit adresátovi. Ten pak tímto klíčem zprávu dešifruje. GDPRCloud momentálně nabízí webové řešení tohoto způsobu zasílání zpráv.

6.2 Architektura webového rozhraní

Pokud se uživatel rozhodne poslat zabezpečený email přes webové rozhraní GDPRCloudu, dostane se na stránku se standardním webovým formulářem.



Obrázek 6.1: GDPRCloud webový formulář pro vyplnění emailu [64]

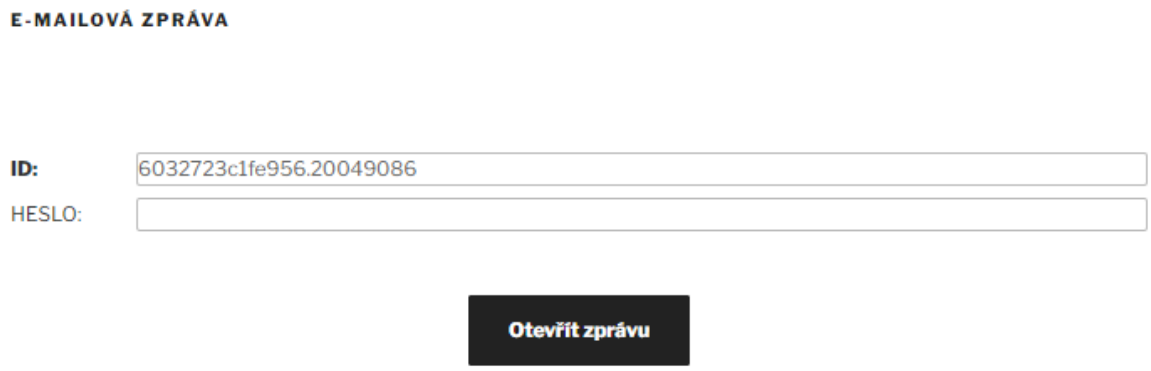
Po vyplnění všech požadovaných kolonek, napsání těla emailu a přiložením případných příloh, musí uživatel napsat popřípadě si nechat vygenerovat heslo. Tímto heslem je pak následně celá zpráva zašifrována. Dále se pak v kolonce „Platnost:“ vybere doba trvanlivosti emailu. To znamená, že po uživatelem definovanou dobu bude možné danou zprávu získat a po vypršení této doby, bude zpráva smazána.

Pro šifrovaní zprávy se používá OpenPGP protokol. Po zašifrování je celá zpráva ode-

slána na server. Pokud uložení na server proběhne v pořádku, server vygeneruje pro daný email ID, které pošle klientovi zpět. Klient poté používá toto ID jako parametr, aby server věděl, že další data se týkají právě dané zprávy. Následně klient pomocí MD5 hashovací funkce vygeneruje otisk (hash) dané zprávy. Tento otisk pak zašifruje stejným heslem, jakým je šifrována celá zpráva. Následně se zašifrovaný i nezašifrovaný hash pošle zpět na server.

Tento celý proces se děje v rámci komunikace klientské strany a GDPRCloud serveru. Uživatel po stisknutí tlačítka odeslat výše popsané kroky neprovádí. Odeslaný email opouští počítač uživatele už zašifrovaný a na serveru, kde čeká na vyzvednutí adresátem, je také zašifrovaný. Server nemá ani informaci o tom, jak má daný email vypadat. V jednotlivých adresářích na serveru je uložená zašifrována zpráva a stejným heslem zašifrované ID.

Server vygeneruje informační email, který zašle adresátovi zprávy. V emailu přijde adresátovi odkaz na server, kde je k vyzvednutí daný email.



Obrázek 6.2: GDPRCloud webový formulář pro přečtení emailu [64]

Při kliknutí odkaz se uživatel dostane na webovou stránku. Na tuto stránku server pošle zašifrovaný hash. Poté, co uživatel vyplní heslo, se pokusí klient tento hash rozšifrovat a rozšifrovaný hash poté pošle zpět na server. Pokud se povedlo daný hash rozšifrovat, server pošle k rozšifrování celý email.

6.3 Proxy

Výše popsané webové řešení ve většině případu nevyhovuje potenciálním uživatelům. Praxí u dnešních uživatelů bývá, že již vlastní nějaký emailový účet, který provozují na desktopovém MUA. Předpokládá se, že pro tyto zákazníky by bylo nepohodlné pro každé odeslání emailu chodit na internetové stránky GDPRCloudu. Navíc se tímto způsobem uživatel obírá o možnost využívat standardní funkce MUA, jako například ukládání emailových adres, ukládání odeslaných emailů na disk atd.

6.3.1 Původní myšlenka

Cílový uživatelé jsou např. malá právnická kancelář do deseti lidí. Tato právnická kancelář používá svého zavedeného mailového klienta (např. Microsoft Outlook nebo Mozilla Thuderbird). Proxy server má tedy zajistit uživatelům aby nadále mohli používat své MUA, ale aby odchozí zprávy odcházeli z kanceláře zašifrované.

Do této kanceláře se umístí vytvořený SMTP proxy server běžící např. na Raspberry PI. Proxy server bude v kancelářské vnitřní síti za firewallem. Z vnitřní sítě na něj bude přístup ze všech PC, ale pouze na povolené porty. Vnitřní síť z hlediska architektury proxy serveru je považována za tzv. demilitarizovanou zónu. To znamená, že uvnitř sítě nehrozí nebezpečí a není potřeba žádného zabezpečení při přenosu mezi jednotlivým subjekty v síti. Přistup z venkovní sítě bude zakázán.

Uživatelé PC v kanceláři vytvoří email, a protože ve svém MUA budou mít nastavený jako výstupní mailový server právě Raspberry PI, tak zpráva půjde jako první na tento server. Ten přijme email, zanalyzuje ho a zprávu zašifruje. Načež ho uloží na GDPRCloud server.

6.3.2 Softwarové specifikace

Celý program je psaný v programovacím jazyce Java. Jedna z výhod Javy je, že je multiplatformní. Program se staví a kompiluje do formátu JAR (Java Archive). Tento JAR soubor může být spuštěn v jakémkoliv operačním systému, pokud v daném OS nainstalována stejná verze JRE (Java Runtime Enviroment). Tento vytvořený program běží na tzv. virtuálním stroji, který je součásti JRE.

Jelikož operační systém v Rassbery PI je postaven na linuxovém jádře, není problém do něj doinstalovat i JRE. Pro implementaci bylo vybráno pro linux defaultní JRE — tedy JRE 11.0.4.

6.3.3 Funkční specifikace

Proxy server je prostředník mezi uživatelem a jím požadovanou destinací – v tomto případě HTTP serveru GDPRCloud. Na proxy server přijde email dle scénáře SMTP protokolu. Během příkazu AUTH dostane proxy server jméno a heslo uživatele. Identita uživatele se ověří přímo na stránkách GDPRCloudu. Jakmile dorazí všechna data a ukončí se komunikace s uživatelem, je celá zpráva zpracována a uložena do bufferu.

Část proxy serveru starající se o odesílání zpráv tento buffer pravidelně kontroluje a odesílá všechny zprávy na GDPRCloud. Nejdříve se zavolá šifrovací služba. Ta v sobě obsahuje generátor hesel a zároveň metody a šifrovaní textu. Vytvořeným nebo uživatelem dodaným heslem je pak celá zpráva zašifrována. Pro šifrovaní zprávy se používá OpenPGP architektura.

Podstatou proxy je, že z jedné strany se pro uživatele tváří z hlediska TCP/IP komunikace jako server a z druhé strany jako klient připojující se na server.

Serverová část proxy je konkrétně SMTP server pro příjem elektronické pošty. Z druhé strany je implementovaný HTTP klient, který komunikuje přes API s cílovou destinací — GDPRCloud serverem.

6.3.4 Bezpečnostní specifikace

U každého emailové zprávy se šifruje pouze tělo zprávy, popř. přílohy. Obálka zůstává z důvodu funkčnosti nezašifrována. Pro zašifrování se používá symetrické šifrování pomocí hesla. Toto heslo buď určí uživatel, nebo uživatel poskytne při odeslání telefonní číslo adresáta. V tom případě heslo generuje proxy server a následné je odesláno pomocí GSM klienta jako SMS adresátovi.

Vygenerované heslo je náhodný řetězec znaků. Požadovaný počet znaků je nastavitelný v konfiguraci proxy serveru. Celá zpráva je následně zašifrovaná algoritmem AES256.

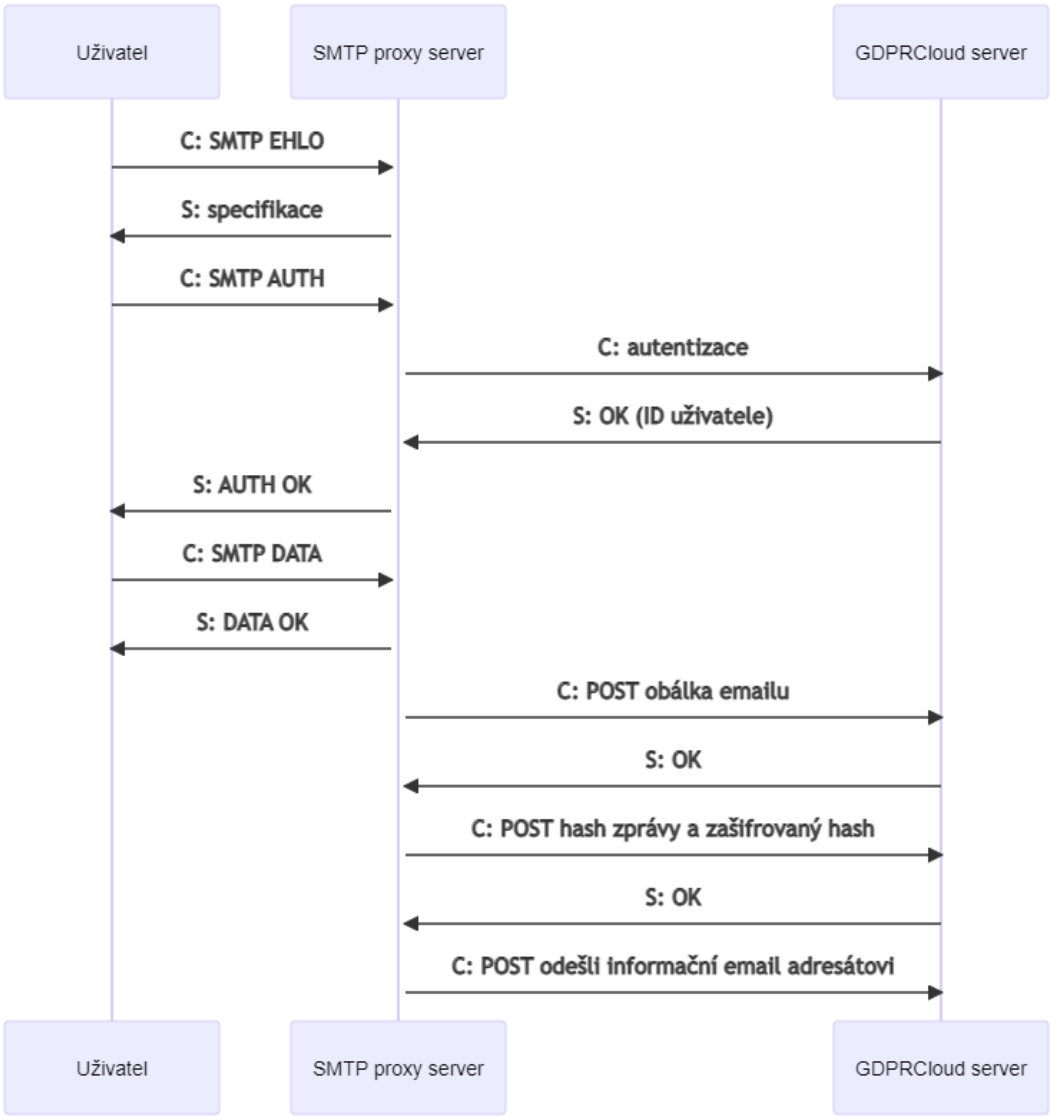
6.4 Architektura softwarového řešení

Z podrobně definovaných specifikacích zadání byla postupně sestavena architektura celého proxy serveru. Ta se v prvé řadě opírala o zadání, v druhé potom o potřeby použitých technik programování, frameworků a knihoven.

6.4.1 Sekvenční diagram

Je pravidlem, že sekvenční diagramy se dělají pro jednotlivé třídy. V tomto případě však byl pro lepší představu posloupnosti událostí vytvořen UML sekvenční diagram pro celý

proces odesílání emailu. Z tohoto diagram se dá jasně vyčíst, co je potřeba implementovat za funkcionality do budoucího modelu tříd.

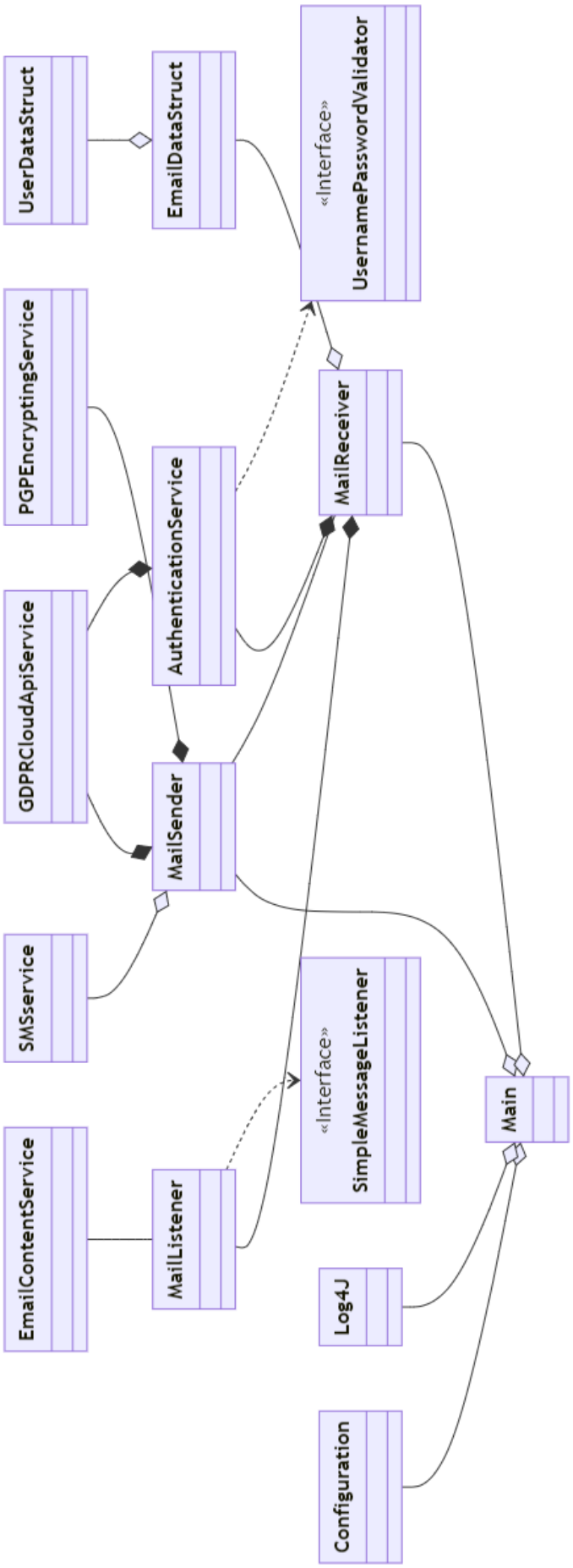


Obrázek 6.3: Sekvenční diagram odeslání emailové zprávy přes proxy server

6.4.2 Třídový model

Jak už bylo zmíněno výše, proxy má dvě základní části — server a klient. A z tohoto pohledu bylo také přistupováno i k tvorbě tříd. Pro příjem zpráv byla vytvořena třída MailReciever a pro odesílání zpráv zase třída MailSender.

Celý diagram tříd včetně metod a atributů byl příliš velký na to, aby se dal umístit na jednu stranu A4. Proto se na obrázku vyskytují pouze samotné třídy. Celý diagram je pak umístěn v přílohách této diplomové práce.



Obrázek 6.4: Třídový diagram

6.4.3 Main

Hlavní třída je třída Main. Hlavní je proto, že se v ní se nachází pro Javu hlavní metoda „public static void main“. Tato hlavní metoda je volána pomocí JRE, která tuto metodu začne vykonávat na virtuálním stroji. V této metodě také probíhá hlavní nekonečná smyčka pro běh programu.

První částí metody je inicializace loggeru. Pro logování je používán Log4j framework. Tento framework spadá pod Apache Software licenci, což znamená, že se může volně používat, upravovat a šířit bez licenčních poplatků.

Po inicializaci jsou vytvořeny objekty MailSender a MailReciever. Poté jsou přes metody „setters“ navzájem předány reference. Po úspěšném předání je zavolána metoda objektu MailReceiver — „serverStart“. Ta obsahuje všechny nutné operace pro spuštění SMTP serveru.

ShutdownHook

Aby bylo v logu poznat, kdy případně byl program ukončen, bylo nutné vytvořit objekt třídy Thread. V konstruktoru této třídy pomocí lambda výrazu napsat požadovanou logovací hlášku. Tento Thread (neboli vlákno) se předá jako parametr runtimové funkci „addShutdownHook“. Shutdown Hooks je speciální konstrukce, která vývojářům umožňuje připojit část kódu, která se má spustit při vypínání JVM.

```
Thread printingHook = new Thread(() -> Log4J.mainProgram.error
("******************** PROGRAM STOPPED ********************"));
Runtime.getRuntime().addShutdownHook(printingHook);
```

Hlavní programová smyčka

Poté už nasleduje hlavní výkoná nekonečná smyčka. Zde se zaprvé kontroluje, zda SMTP server běží a v případě, že neběží, je opět nastartován a za druhé se volá v MailSenderu hlavní výkoná metoda, která má za úkol zprocesovat všechny přijaté maily od MailReceivera uložené v bufferu. Celá tato nekonečná smyčka je v bloku „try-catch“ k odchytnutí a uložení chyby do logu.

```
try
{
    //MAIN PROGRAM LOOP
    while(true) {
        if(mailReceiver.getSmtpServer().isRunning()){
            Log4J.receiverLogger.debug("SMTP server is running...");
        }
        else {
            mailReceiver.serverStop();
            mailReceiver.serverStart();
        }

        mailSender.processEmails();
        Thread.sleep(ProgramCommons.MAILSENDER_PROCESS_PERIOD);
    }
}
catch (Exception e){
    Log4J.mainProgram.error("Program Exception: {}", e);
}
```

6.4.4 MailReceiver

Tato třída obsahuje všechny důležité atributy pro zprovoznění SMTP serveru. Samotný server nebylo potřeba definovat od základu, protože byla použita knihovna Subethamail. Ta obsahuje třídu SMTP server. Tato třída poskytuje veškeré potřebné komponenty, pro vytvoření TCP/IP serveru, který právě rozumí SMTP příkazům.

Ve třídě MailReceiver je metoda pro nastartování serveru a metoda pro vypnutí serveru. Metoda v prvé řadě kontroluje, zda vůbec existuje instance třídy SMTPServer. Ta se vždy vytváří jako atribut třídy MailReceiver. Následně se pro zastavení kontroluje, pokud daný server vůbec běží.

Pro nastartovaní serveru, se volá metoda start. Ta vytváří instanci výše zmiňované třídy SMTPServer. Tato instance potřebuje několik parametrů dle definice z dokumentace knihovny.

Server se rozběhne ve vlastním vlákně. Jedná se o synchronizovanou metodu. Synchronizovaná metoda slouží k uzamčení objektu pro jakýkoli sdílený prostředek. Když vlákno vyvolá synchronizovanou metodu, automaticky získá zámek pro danou instanci a uvolní jej, jakmile vlákno dokončí svou úlohu.

MailListener

V první řadě je to tzv. posluchač zpráv (MessageListener). Tento posluchač funguje něco jako vyřizovatel příchozí zprávy. Myšlenka Subethamail knihovny byla taková, že SMTPServer může mít takovýchto posluchačů několik. Je na uživateli, kolik tříd posluchačů s implementací tohoto interfacu si vytvoří.

Třída musí u sebe implementovat dvě metody. První metoda „accept“ se volá, když je během relace s klientem poslán příkaz *MAIL FROM* a *RCPT TO*. V této metodě jsou jako parametry předány adresy odesílatele a příjemce. Při každé nové příchozí zprávě jsou osloveni všichni posluchači, zda daný email chtějí přijmout či nikoliv.

Vytvořený proxy server má pouze jednu instanci posluchače. Jelikož předpoklad prostředí tohoto serveru je tzv. demilitarizovaná zóna, tak jediná podmínka, zda zpráva má či nemá být serverem přijata je nevyplněný odesílatel nebo doručovatel.

Na to přímo navazuje druhá metoda „delivered“, která je zavolána v momentě, kdy server přijme celou zprávu. Jako parametr této funkce je předána objekt typu Inputstream. Přijatou zprávu je tedy potřeba zanalyzovat a uložit do nějaké datové skruktury. Z tohoto důvodu byla vytvořena třída EmailDataStruct. Ta obsahuje všechny atributy potřebné pro správně fungování přeposílání zpráv. V metodě delivered se tedy pro každou zprávu vytvoří instance této třídy.

EmailContentService

Tato třída slouží pro analyzování a třídění dat přijatých SMTPServerem. Nachází se v ní dvě statické metody. Volat se dá pouze hlavní analyzující metoda (je typu *public*). Druhá metoda je pouze pomocná. Funkce hlavní metody je převést přijatý datový typ Inputstream, ve kterém je celý obsah přijaté zprávy do ojektu EmailDataStruct. Statické jsou z důvodu, že tato třída neobsahuje žádné atributy a proto není důvod vytvářet její instanci.

Velmi důležitá část zanalyzování těla zprávy je její sestavení uživatelem před odeslánním. Aby další potřebné funkce fungovali, je potřeba vložit do těla zprávy GDPRCloud hlavičku zprávy. Ta informuje proxy server, jak s danou zprávou naložit. V hlavičce se nachází

- telefonní číslo příjemce zprávy (Pokud není požadováno odeslání hesla SMS zprávou, není tato část vyplněna.)
- heslo těla zprávy — pokud není vyplněno, proxy server vygeneruje vlastní heslo

AuthValidator

Druhý požadovaný parametr pro třídu SMTPServer je nějaký autentizační handler. I v tomto případě se jedná o implementaci interfacu z knihovny. Tento interface obsahuje pouze jednu metodu — „login“. Jako parametr je předáno jméno a heslo.

Jelikož se jedná o proxy server, autentizace je ověřována na cílovém serveru. Tato třída má jako atributy referenci na posluchače, kterou dostane v konstruktoru. Dále si nainstancuje GDPRCloudApiService, což je třída, která má na starosti, jak už název napovídá, komunikaci s GDPRCloud serverem.

Když je teda metoda „login“ zavolána, je uvnitř zavolána metoda třídy GDPRCloudApiService — „authenticate“. Jako parametry jsou jí předány přijaté jméno a heslo. Ta pošle https požadavek k ověření na server. V případě, že je ověření úspěšné, předá se jméno, heslo a nově získané autentizační ID posluchači. Ten tyto údaje uloží do příslušné instance mailové šablony. Pokud ověření je neúspěšné, metoda vyhodí příslušnou chybovou hlášku. Pak je email odmítnut a celá relace je ukončena.

6.4.5 MailSender

Jedná se druhou hlavní část proxy serveru. Má na starosti šifrování a odesílání zpráv na GDPRCloud server. Hlavní výkoná metoda třídy je „processEmails“. Podstatou této metody zkontrolování emailové bufferu a v případě výskytu neodeslaných emailů jsou tyto zprávy zprocesovány.

Tato třída dostává v konstruktoru referenci na instanci MailListener. Je to z důvodu, že právě v tomto objektu se nachází emailový buffer. Tato třída si vytváří instance GDPRCloudApiService a pokud je požadováno, tak i instanci SMSService..

Případ, kdy se třída MailSender pokusí odeslat zpracovaný email na server GDPRCloud. Email je šifrovaný heslem dle OpenPGP protokolu. MailSender požádá PGPEncryptingService pro vygenerování hesla o délce definované v konfiguračním souboru. Po vygenerování hesla je zavolána metoda ze stejné služby, která zašifruje tělo přiložené emailové zprávy právě vygenerovaným heslem.

Po zašifrování je celá zpráva odeslána na server. To zajišťuje instance GDPRCloudService. Pokud celý proces uložení zprávy na server proběhne v pořádku, nastávají 2 možnosti.

1. Z konfigurace bylo vyčteno, že se požaduje použití SMS služby. V tom případě se očekává, že na začátku každého emailu bude napsáno číslo adresáta oddělené středníkem. Pokud tak bylo učiněno, je zavolána právě SMS služba, která zašle na přiložené číslo heslo pro daný email.

2. SMS služba není požadována a proto bylo na začátku emailové zprávy přiloženo heslo, kterým se má daná zpráva zašifrovat.

Pokud se nepodařilo odeslat zprávu na server, je zavolána metoda GDPRCloudApiServicu, aby poslala odesílateli informaci o nezdaření odeslání zprávy z proxy serveru.

```
public void processEmails()
{
    if (this.mailListener.getReceivedEmailsList().size() > 0)
    {
        // loop through emailBuffer
        for (EmailDataStruct email: this.mailListener.getReceivedEmailsList())
        {
            if(email.getBodyPassword().length() == 0){
                //generate  password for encrypting the message
                email.setBodyPassword(PGPEncryptingService
                .generatePassword(ProgramCommons.MAIL_PASSWORD_LENGTH));
            }

            //encrypt email body
            email.setBody(PGPEncryptingService
            .encryptMessage(email.getBody(), email.getBodyPassword()));
            try{
                PGPEncryptingService.countBodyHash(email);
                PGPEncryptingService.encryptBodyHash(email);
            }
            catch (NoSuchAlgorithmException e){
                Log4J.senderLogger.error(e);
            }


            boolean sendingResult = this.apiService.sendMailMainMethod(email);
            if(!sendingResult){
                apiService.sendBounceEmail(email);
            }
            else if (sendingResult
                    && this.smsService != null
                    && !email.getRecipientPhoneNumber().equals("")){
```

```
                //sending the password to the recipient
                smsService.sendSMS(
                                email.getUserData().getUserEmailAddress(),
                                email.getRecipientPhoneNumber(),
                                email.getBodyPassword()
                );
            }
            email.setSentToServer(sendingResult);
        }
        this.mailListener.getReceivedEmailsList()
        .removeIf(x -> x.isSentToServer());
    }
}
```

PGPEncryptingService

Tato třída slouží jako služba pro zajištění bezpečnosti odchozích zpráv. Z prvé jsou zde metody pro generování hesla a šifrování dat pomocí tohoto hesla. Pro šifrování byla použita PGPPainless knihovna, která poskytuje veškeré potřebné mechanismy pro šifrování dle standardu OpenPGP.

Za druhé se zde nachází také metoda pro vytvoření hashe z těla emailové zprávy a metoda pro zašifrování tohoto hashe stejným heslem, jaké bylo použito pro šifrování těla zprávy.

Jelikož tato třída neobsahuje žádné atributy, jsou všechny metody statické. Není potřeba tedy vytvářet instanci této třídy pro zavolání jejích metod.

GDPRCloudApiService

Tato služba byla již mnohokrát zmíněna. Jedná se o komunikační prostředek s GDPRCloud serverem a jeho službami. V první řadě je to HTTP(s) klient. Je to z důvodu, že server poskytuje HTTP API. Pokud se to vezme dle harmonogramu přeposílání zpráv z proxy serveru, tak první HTTP volání je ověření uživatele pomocí jména a hesla. Pokud ověření proběhne v pořádku, je serverem vráceno speciální ID, přes které se dá přistupovat k dalším komunikačním bodům.

Pro odeslání emailu se v této třídě nachází jedna hlavní metoda, který celý scénář zprocesuje. Nejedná se tedy pouze o jeden požadavek. Aby byla celá zpráva uložena na serveru, je potřeba provést 4 požadavky, kdy jsou postupně jednotlivé části emailu předány serveru.

1. sendNewMessage — Zahájení odeslání emailové zprávy. V parametrech volání se zasílá hlavička emailové zprávy.

2. sendFile — Pokud emailová zpráva obsahuje nějaké přílohy, jsou poslány v tomto volání.

3. sendMail — Zasílá se zašifrované tělo zprávy. (Bez GDPRCloud hlavičky. Ta sloužila pouze proxy serveru).

4. sendMailDo — Jedná se poslední volání, který ukončuje celý proces ukládání dat.

Kromě HTTP volání na server se zde také nachází emailový klient, který posílá emailové zprávy jménem GDPRCloudu případné potíže při přeposílání zpráv.

SMSService

Pro odeslání SMS byla využita služba Twillio. Ta nabízí různé služby v oblasti GSM komunikací — v případě tohoto proxy serveru zasílání SMS zpráv. Výhodou této služby je jejích API, které nabízí v rámcí Twillio knihovny i pro jazyk Java.

Aby celá služba mohla fungovat, byl vytvořen na stránkách společnosti Twillio zkušební účet. Twillio uživateli přidělí SID, autorizační token. Tyto údaje se pak používají k autentizaci přístupu přes API. Uživateli je pak přiděleno telefonní číslo, ze kterého budou chodit případné zprávy požadovaným uživatelům.

Díky právě vytvořenému API je celý proces odesílání zpráv velmi jednoduchý. Pro tuto aplikaci musela být využita starší Twillio knihovna, jelikož nejnovější verze nefungovala. Jeden z možných důvodu je, že celý program je stavěn na starším JDK 11, kde můžou scházet některé požadované funkce.

6.5 Nastavení MUA

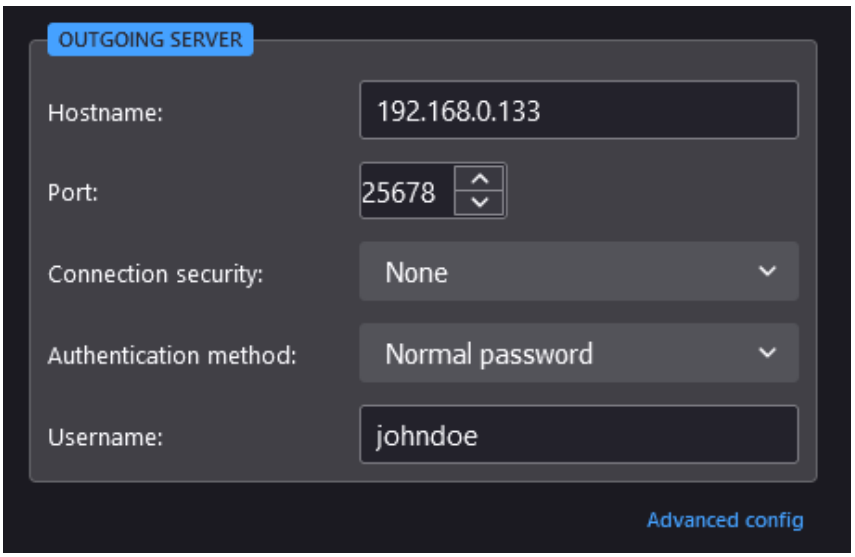
Jako MUA byla zvolena Mozzilla Thunderbird. Jedná se o volně dostupného emailového klienta. Obdobný postup by se uplatnil i u jiných klientů.

6.5.1 Nasměrovaní odchozí pošty na proxy server

V první řadě je potřeba vytvořit uživatele. Ten se vytváří standardním způsobem a proto zde daný způsob nebude popsán. Co je ale důležité, tak při tomto nastavení kliknout na „Nakonfigurovat manuálně“. Jedná se o nastavení serverů pro odchozí a příchozí poštu. Ten pro odchozí není důležitý, jelikož maily odeslané přes proxy nebudou uloženy na žádný server, ale budou uloženy lokálně na vašem počítači (popř. budou rovnou smazány).

Server pro odchozí poštu je potřeba nastavit, aby odchozí pošta chodila právě na proxy server. Do kolonky „Host name“ se vyplní IP adresa počítače, kde běží daný proxy server. Port se nastavený stejný, jako je v konfiguračním souboru proxy serveru. Uživatelské jméno se nastaví stejné, jaké má uživatel na GDPRCloudu.

Co se týče bezpečnosti připojení, tak se nastaví na „Žádné“. Metoda autentizace je zvolena „heslo“. Tímto se zajistí, že proxy server bude mít potřebné údaje pro ověření identity uživatele na GDPRCloud serveru.



Obrázek 6.5: Příklad nastavení serveru odchozí pošty v aplikaci Thunderbird

6.5.2 Vytvoření zprávy

Ke svému účtu je potřeba přiložit šablonu emailu. Tato šablona bude nastavena jako výchozí — při každém psaní nového emailu bude použita tato šablona. Podstatnou části je, že v těle zprávy se bude nacházet zmiňovaná hlavička pro proxy server.

V ní musí uživatel vyplnit buď telefonní kontakt adresáta nebo heslo (popř obojí), kterým má být daná zpráva zašifrována. Pokud nebude vyplněn ani jedna z možností nebo budou vyplněna nesprávně (např. špatná délka nebo tvar telefonního čísla), zpráva nebude předána dál a bude v proxy serveru smazána.

Příklad, jak takové tělo zprávy vypadá:

```
******BEGIN PROXY HEADER*****
RecipientPhoneNumber:+420123456789<End>
EmailBodyPassword:password<End>
******END PROXY HEADER*****

Dále pokračuje uživatelem napsaná zpráva.
```

Kapitola 7

Závěr

Cílem této diplomové práce bylo vytvoření proxy serveru pro šifrování odchozí pošty. Jedna z cílových předností proxy serveru je zajištění end-to-end šifrování s minimálním zásahem běžného uživatele.

Pro vytvoření tohoto serveru jsem se musel seznámit s architekturou emailové komunikace, bezpečnosti emailové pošty a možnostmi v oblasti proxy serverů. Na základě toho jsem v teoretické části provedl rešerši na všechny zmíněné oblasti.

Jako první jsem se zaměřil na jednotlivé moduly emailové komunikace - tzv. agenty. Popsal jsem význam jednotlivých agentů a jejich funkce v emailové architektuře.

Dále bylo nezbytné seznámit se s komunikačními protokoly elektronické pošty. Základním standardem pro přenos pošty je protokol SMTP. Jeho jednotlivé části a způsob fungování jsem rozebral ve druhé kapitole. Stejně tak jsem provedl rešerši na protokoly pro příjem a načítání zpráv. Jedná se o protokoly POP3 a IMAP. V práci, kromě popisu samotných protokolů, také zmiňuji výhody a nevýhody obou variant.

Jelikož základní myšlenkou praktické části bylo zlepšení zabezpečení odesílaných dat, je další část věnována bezpečnosti. V případě bezpečnosti přenosu jsem popsal prokol TLS, jehož využití v emailové komunikaci je dnes již standardem. Pro docílení end-to-end šifrování jsem zvolil a popsal nejrozšířenější standard pro elektronickou poštu - OpenPGP.

V poslední kapitole teoretické části se věnuji rešerši v oblasti proxy serverů. Zaměřuji se na definici proxy serveru, jeho typy a již existující řešení v oblasti SMTP proxy serverů.

Hlavním výstupem mé diplomové práce je naprogramované řešení SMTP proxy serveru, který šifruje odchozí poštu. Architektura řešení se opírá o nabyté znalosti z teoretické části diplomové práce. Zároveň bylo celé řešení přizpůsobeno již existujícímu řešení ve službě GDPRCloud, kterého je součástí.

Architekturu vytvořeného proxy serveru, včetně softwarového řešení, jsem popsal v kapitole 6. Tento server má všechny potřebné funkcionality pro úspěšné odeslání zašifrované zprávy na GDPRCloud. Proxy server poskytuje symetrické šifrování pomocí hesla.

Toto heslo si uživatel může zvolit nebo poskytne v emailu telefonní číslo adresáta. V tomto případě proxy server heslo vygeneruje a pošle adresátovi SMS zprávu s heslem. Jeho funkčnost jsem experimentálně ověřil.

7.1 Případná vylepšení

Jedná se o první verzi tohoto proxy serveru, takže prostor pro možná zlepšení je vysoce pravděpodobný. Jedna věc je zlepšení samotného kódu vytvořeného proxy serveru a druhá věc je vylepšení celé architektury řešení.

7.1.1 Architektura řešení

Co se týče architektury, tak zde by určitě pomohla naprogramovat rozšíření pro MUA. Jak MS Outlook, tak Mozzila Thunderbird nabízejí za určitých podmínek možnost naprogramovat vlastní rozšíření. Takové rozšíření by pomohlo jak po estetické stránce, kde by se například hlavička proxy serveru mohla být schována do nějakého grafického rozhraní. Dále by pak mohla šifrovat zprávy a proxy by pak už fungovala spíše jako překladač protokolů ze SMTP do HTTP(s). Tím by se docílilo skutečného end-to-end šifrování.

Dále pak vylepšení metody zasílání zpráv. Pro zasílání SMS se počítá pouze s jedním adresátem. Pokud by se tímto způsobem měla zasílat SMS zpráva více uživatelům, bylo by potřeba hlavičku zrevidovat. Při zasíláním vícero adresátům by se pro uživatele stávala zpráva značně nepřehledná. Toto by opět zlepšilo grafické rozhraní. Pro ověření funkčnosti celé architektury byl použit Twillio provider. Toto řešení však není zdarma. Možné řešení je například kombinace Rasspberry PI Arduina, které nabízí potřebný GSM modul.

7.1.2 Kód programu

Samotný kód programu by se také dál určitě vylepšit. V první řadě je to zlepšení robustnosti celého běhu programu. Hodně chybových hlášek bylo ošetřeno try-catch blokama. Ty které způsobí fatální problém (jako např. nenalezení konfiguračního souboru) zastaví celý program. Kontrola získaných dat by se dala určitě zlepšit a zpřísnit pomocí regulárních výrazů.

Dále pak je to uschování potřebných bezpečnostních údajů. Momentálně se nacházejí v samotném zdrojovém kódu, což představuje potenciální riziko.

Bibliografie

[1] M. T. Banday, J. Qadri a N. Shah, "A Practical Study of E-mail Communication through SMTP," květ. 2010.

[2] J. Scheerder a C. Koymans, "Email," led. 2007. DOI: 10.1016/B978-044452198-9.50009-4.

[3] T. Holland, *Webmail vs email clients: Which one should you choose?* 2018. URL: https://www.ontrack.com/en-us/blog/webmail-vs-email-clients-one-choose.

[4] *Webmail vs. email client*. URL: https://www.doteasy.com/domain-email-and-website-hosting-articles/webmail-vs-email-client.

[5] *What is Webmail - javatpoint*, en. URL: https://www.javatpoint.com/what-is-webmail (cit. 02. 04. 2022).

[6] *Mail retrieval agent*, en. URL: https://en-academic.com/dic.nsf/enwiki/6283229 (cit. 01. 03. 2022).

[7] *Mail Transfer Agent (MTA) Explained | Mailtrap*, en-GB, Section: Email Infrastructure, říj. 2019. URL: https://mailtrap.io/blog/mail-transfer-agent/ (cit. 01. 03. 2022).

[8] *What is MTA - Javatpoint*, en. URL: https://www.javatpoint.com/what-is-mta (cit. 02. 04. 2022).

[9] *How Does Mail Routing Work? - Linux Network Administrator's Guide, Second Edition [Book]*, en, ISBN: 9781565924000. URL: https://www.oreilly.com/library/view/linux-network-administrators/1565924002/ch17s04.html (cit. 02. 04. 2022).

[10] J. C. Klensin a R. Gellens, "Message Submission for Mail," Internet Engineering Task Force, Request for Comments RFC 6409, lis. 2011, Num Pages: 20. DOI: 10.17487/RFC6409. URL: https://datatracker.ietf.org/doc/rfc6409 (cit. 02. 04. 2022).

[11] *Mail delivery agent*, en. URL: https://en-academic.com/dic.nsf/enwiki/153371 (cit. 02. 04. 2022).

[12] *Topic - Message Access Agent (MAA)*, en. URL: https://pdfcoffee.com/topic-message-access-agent-maa-pdf-free.html (cit. 02. 04. 2022).

[13] S. Ramadass, H. Bazar, O. Abouabdalla a A. Manasrah, "Active E-mail system protocols monitoring algorithm," lis. 2009. DOI: 10.1109/TENCON.2009.5396120.

[14] J. C. Klensin, "Simple Mail Transfer Protocol," Internet Engineering Task Force, Request for Comments RFC 2821, dub. 2001, Num Pages: 79. DOI: 10.17487/RFC2821. URL: https://datatracker.ietf.org/doc/rfc2821 (cit. 03. 04. 2022).

[15] P. Resnick, "Internet Message Format," Internet Engineering Task Force, Request for Comments RFC 2822, dub. 2001, Num Pages: 51. DOI: 10.17487/RFC2822. URL: https://datatracker.ietf.org/doc/rfc2822 (cit. 03. 04. 2022).

[16] *Understanding an email header | Media Temple Community*. URL: https://mediatemple.net/community/products/dv/204643950/understanding-an-email-header (cit. 03. 04. 2022).

[17] Archiveddocs, *MIME Message Format*, en-us. URL: https://docs.microsoft.com/en-us/previous-versions/office/developer/exchange-server-2010/aa494197(v=exchg.140) (cit. 03. 04. 2022).

[18] N. Freed a N. S. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," Internet Engineering Task Force, Request for Comments RFC 2045, lis. 1996, Num Pages: 31. DOI: 10.17487/RFC2045. URL: https://datatracker.ietf.org/doc/rfc2045 (cit. 03. 04. 2022).

[19] ——, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," Internet Engineering Task Force, Request for Comments RFC 2046, lis. 1996, Num Pages: 44. DOI: 10.17487/RFC2046. URL: https://datatracker.ietf.org/doc/rfc2046 (cit. 03. 04. 2022).

[20] K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text," Internet Engineering Task Force, Request for Comments RFC 2047, lis. 1996, Num Pages: 15. DOI: 10.17487/RFC2047. URL: https://datatracker.ietf.org/doc/rfc2047 (cit. 03. 04. 2022).

[21] J. Postel, J. C. Klensin a N. Freed, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures," Internet Engineering Task Force, Request for Comments RFC 2048, lis. 1996, Num Pages: 21. DOI: 10.17487/RFC2048. URL: https://datatracker.ietf.org/doc/rfc2048 (cit. 03. 04. 2022).

[22] N. Freed a N. S. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples," Internet Engineering Task Force, Request for Comments RFC 2049, lis. 1996, Num Pages: 24. DOI: 10.17487/RFC2049. URL: https://datatracker.ietf.org/doc/rfc2049 (cit. 03. 04. 2022).

[23] S. Ramadass, H. Bazar, O. Abouabdalla a A. Manasrah, "Active E-mail system protocols monitoring algorithm," lis. 2009. DOI: 10.1109/TENCON.2009.5396120.

[24] J. C. Klensin, "Simple Mail Transfer Protocol," Internet Engineering Task Force, Request for Comments RFC 5321, říj. 2008, Num Pages: 95. DOI: 10.17487/RFC5321. URL: https://datatracker.ietf.org/doc/rfc5321 (cit. 02. 04. 2022).

[25] *What is a DNS MX record?* URL: https://www.cloudflare.com/learning/dns/dns-records/dns-mx-record/.

[26] S. Redin, *The SMTP Protocol Fundamentals*, en, 2013. URL: https://en.redinskala.com/the-smtp-protocol-fundamentals/ (cit. 01. 03. 2022).

[27] *DNS a record - cloudflare*. URL: https://www.cloudflare.com/learning/dns/dns-records/dns-a-record/.

[28] V. Riabov, "SMTP (Simple Mail Transfer Protocol)," in pros. 2007, s. 388–406, ISBN: 978-0-471-78459-3. DOI: 10.1002/9781118256114.ch26.

[29] M. T. Banday, J. Qadri a N. Shah, "A Practical Study of E-mail Communication through SMTP," květ. 2010.

[30] *SMTP Commands Reference (covers HELO/EHLO, MAIL, RCPT, DATA, RSET, VRFY, AUTH, STARTTLS etc)*. URL: https://www.samlogic.net/articles/smtp-commands-reference.htm (cit. 03. 04. 2022).

[31] B. Specht, *Everything you need to know about SMTP (simple mail transfer protocol)*, 2021. URL: https://postmarkapp.com/guides/everything-you-need-to-know-about-smtp.

[32] R. Siemborski a A. Melnikov, "SMTP Service Extension for Authentication," Internet Engineering Task Force, Request for Comments RFC 4954, čvc. 2007, Num Pages: 20. DOI: 10.17487/RFC4954. URL: https://datatracker.ietf.org/doc/rfc4954 (cit. 03. 04. 2022).

[33] *POP Protocol | Post Office Protocol - javatpoint*, en. URL: https://www.javatpoint.com/pop-protocol (cit. 03. 04. 2022).

[34] D. Grignani, *Understanding Post Office Protocol (POP3)*. URL: https://www.2brightsparks.com/resources/articles/understanding-post-office-protocol-pop3.html (cit. 04. 03. 2022).

[35] J. Myers a M. Rose, "Post Office Protocol - Version 3," en, RFC Editor, tech. zpr. RFC1939, květ. 1996, RFC1939. DOI: 10.17487/rfc1939. URL: https://www.rfc-editor.org/info/rfc1939 (cit. 05. 06. 2022).

[36] C. Chung, *Receiving Email with Internet Message Access Protocol (IMAP4)*, en. URL: https://www.2brightsparks.com/resources/articles/internet-message-access-protocol.html (cit. 04. 03. 2022).

[37] A. Melnikov a B. Leiba, "Internet Message Access Protocol (IMAP) - Version 4rev2," Internet Engineering Task Force, Request for Comments RFC 9051, srp. 2021, Num Pages: 163. DOI: 10.17487/RFC9051. URL: https://datatracker.ietf.org/doc/rfc9051 (cit. 05. 06. 2022).

[38] L. Cailleux, A. Bouabdallah a J.-M. Bonnin, "Building a Confident Advanced Email System Using a New Correspondence Model," květ. 2014, s. 85–90, ISBN: 978-1-4799-2653-4. DOI: 10.1109/WAINA.2014.24.

[39] R. Shukla, H. O. Prakash, R. P. Bhushan, S. Venkataraman a G. Varadan, "Sahastradhara: Biometric and EToken integrated secure email system," in *2013 15th International Conference on Advanced Computing Technologies (ICACT)*, 2013, s. 1–4. DOI: 10.1109/ICACT.2013.6710516.

[40] A. Humadi, "Symmetric and Asymmetric Encryption," pros. 2020. DOI: 10.13140/RG.2.2.21500.56962.

[41] M. Tracy, W. Jansen, K. Scarfone a T. Winograd, "NIST Special Publication 800-44 Version 2, Guidelines on Securing Public Web Servers," zář. 2007.

[42] C. Allen a T. Dierks, "The TLS Protocol Version 1.0," Internet Engineering Task Force, Request for Comments RFC 2246, led. 1999, Num Pages: 80. DOI: 10.17487/RFC2246. URL: https://datatracker.ietf.org/doc/rfc2246 (cit. 04. 04. 2022).

[43] *SSL/TLS Best Practices for 2021*, en-US. URL: https://www.ssl.com/guide/ssl-best-practices/ (cit. 04. 04. 2022).

[44] *Secure Socket Layer (SSL)*, en-us, Section: Computer Networks, čvn. 2019. URL: https://www.geeksforgeeks.org/secure-socket-layer-ssl/ (cit. 04. 04. 2022).

[45] *TLS Security 3: SSL/TLS Terminology and Basics*, en-US, břez. 2019. URL: https://www.acunetix.com/blog/articles/tls-ssl-terminology-basics-part-3/ (cit. 05. 04. 2022).

[46] *TLS Security 5: Establishing a TLS Connection*, en-US, břez. 2019. URL: https://www.acunetix.com/blog/articles/establishing-tls-ssl-connection-part-5/ (cit. 05. 04. 2022).

[47] E. Rescorla a T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Engineering Task Force, Request for Comments RFC 5246, srp. 2008, Num Pages: 104. DOI: 10.17487/RFC5246. URL: https://datatracker.ietf.org/doc/rfc5246 (cit. 05. 04. 2022).

[48] *About*, en, led. 2022. URL: https://www.openpgp.org/about/ (cit. 10. 04. 2022).

[49] *What is PGP Encryption and How Does It Work?* en. URL: https://www.varonis.com/blog/pgp-encryption (cit. 10. 04. 2022).

[50] *What is Pretty Good Privacy and how does it work?* en. URL: https://www.techtarget.com/searchsecurity/definition/Pretty-Good-Privacy (cit. 10. 04. 2022).

[51] H. Finney, L. Donnerhacke, J. Callas, R. L. Thayer a D. Shaw, "OpenPGP Message Format," Internet Engineering Task Force, Request for Comments RFC 4880, lis. 2007, Num Pages: 90. DOI: 10.17487/RFC4880. URL: https://datatracker.ietf.org/doc/rfc4880 (cit. 10. 04. 2022).

[52] R. M. l. updated, *Best secure email providers of 2022*, en, ún. 2022. URL: https://www.techradar.com/best/best-secure-email-providers (cit. 28. 05. 2022).

[53] S. Kozak, M. Murin a W. Wei, "PreVeil E2EE Email: Security Review," en, s. 5,

[54] K. A. l. updated, *SecureMyEmail secure email review*, en, lis. 2020. URL: https://www.techradar.com/reviews/securemyemail (cit. 28. 05. 2022).

[55] *SecureMyEmail - Encrypted Email Security Practices*, en-US. URL: https://www.securemyemail.com/encrypted-email-security/ (cit. 28. 05. 2022).

[56] *Proton is privacy – True privacy without compromise.* URL: https://proton.me/privacy (cit. 28. 05. 2022).

[57] R. S. l. updated, *ProtonMail secure email review*, en, lis. 2020. URL: https://www.techradar.com/reviews/protonmail-secure-email (cit. 28. 05. 2022).

[58] M. Sysel a O. Doležal, "An Educational HTTP Proxy Server," *Procedia Engineering*, roč. 69, 128–132, pros. 2014. DOI: 10.1016/j.proeng.2014.02.212.

[59] *What is a SOCKS5 proxy and why should you use one? | NordVPN*, en, břez. 2022. URL: https://nordvpn.com/blog/socks5-proxy/ (cit. 28. 05. 2022).

[60] A. Frisch, *Essential system administration*, 3rd ed. Beijing ; Sebastopol, CA: O'Reilly, 2002, ISBN: 978-0-596-00343-2.

[61] *SMTP Proxy.* URL: https://docs.oracle.com/cd/E19047-01/sunscreen32/806-6347/6jfa0g88r/index.html (cit. 28. 05. 2022).

[62] C. Research, *What is SMTP tarpitting? What are SMTP tarpits (or tar pits)?* en. URL: https://verifalia.com/help/email-validations/what-is-smtp-tarpitting (cit. 28. 05. 2022).

[63] *Leveraging Outbound SMTP Proxy Server Can Protect Your Organization From Blacklisting - DuoCircle*, en-US. URL: `https://www.duocircle.com/content/outbound-spam-filtering/outbound-smtp-proxy` (cit. 28. 05. 2022).

[64] *GDPR cloud – Služby kybernetické bezpečnosti a GDPR*, cs. URL: `http://gdprcloud.cz/` (cit. 31. 05. 2022).

[65] *How does the GDPR affect email?* en-US, Section: GDPR Compliance, čvc. 2018. URL: `https://gdpr.eu/email-encryption/` (cit. 31. 05. 2022).