VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Matyáš Horký

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

OVĚŘOVÁNÍ DIGITÁLNÍCH PODPISŮ SYSTÉMU PGP
VERIFICATION OF PGP DIGITAL SIGNATURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matyáš Horký**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Václav Zeman, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Matyáš Horký ***ID:*** 211790
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Ověřování digitálních podpisů systému PGP

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit knihovnu pro Python podporující ověřování digitálních podpisů e-mailové komunikace dle standardu PGP RFC 3156. Knihovna umožní snadným způsobem rozdělení těla zprávy e-mailu na podepsanou a nepodepsanou část a ověření textu zprávy a příloh. Výstupem bude objekt, který by obsahoval všechna získaná data, od použitého klíče po obsah samotný.

DOPORUČENÁ LITERATURA:

[1] RFC3156 MIME Security with OpenPGP https://tools.ietf.org/html/rfc3156

[2] Mark Summerfield, Python 3, ISBN: 978-80-251-2737-7 Brno, Computer Press, 2010

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zaměřuje na práci s OpenPGP obsahem v e-mailových souborech. Prochází historii, změny a vlastnosti PGP a e-mailu a zasazuje je do společného kontextu. V rámci této práce je navrhnut a implementován program pro čtení a ověřování OpenPGP dat v elektronické poště, který je dostupný jako open-source nástroj pro použití z příkazové řádky i ve funkci knihovny.

KLÍČOVÁ SLOVA

e-mail, EFAIL, GnuPG, OpenPGP, PGP, Python, S/MIME, SMTP

ABSTRACT

This bachelor thesis is focused on the interaction of OpenPGP content inside of e-mail files. It describes the history, changes and properties of PGP and e-mail and connects them into a common context. The goal is to design and implement a program for reading and verifying OpenPGP data in electronic mail, which is available as an open-source tool available as both a the command line and as a library.

KEYWORDS

e-mail, EFAIL, GnuPG, OpenPGP, PGP, Python, S/MIME, SMTP

HORKÝ, Matyáš. *Ověřování digitálních podpisů systému PGP*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 61 s. Bakalářská práce. Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Matyáš Horký
VUT ID autora:	211790
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Ověřování digitálních podpisů systému PGP

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce panu doc. Ing. Václavu Zemanovi, Ph.D. za přijetí vlastního tématu práce a jeho odborné vedení. Ivaně Fitere za podporu a možnost využít její jméno v demonstračních ukázkách. Janě Kotherové za korektury a cenné komentáře k textu.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V létě 2020 jsem se v rámci stáže dostal k nástroji `intelmq`. Tento program byl navržen a vytvořen Evropskými CERT (*Community Emergency Response Team*) a CSIRT (*Computer Security Incident Response Team*) týmy za účelem sběru a využívání dat bezpečnostního charakteru (logy, upozornění a podobně) [1].

Samotný program tvoří mnoho malých, specializovaných *botů*, kteří sbírají, sjednocují, analyzují a ukládají přijatá data. Sběrači se starají o přidávání nových logů a `http collector` má od verze 2.3.0 podporu ověřování PGP podpisů souborů [2]. Bot `mail collector` takové ověřování nemá; to mě přivedlo na myšlenku vytvořit knihovnu, která toho bude schopná.

E-mail je základní a univerzální způsob komunikace přes internet. V počátcích internetu nebyla bezpečnost prioritou, e-mail je tedy stejně jako většina původních internetových protokolů (http, ftp, telnet) služba nezabezpečená a data se po síti přenášejí v otevřené podobě.

Využití PGP (*Pretty Good Privacy*) zajišťuje autenticitu a bezpečnost přenášeného obsahu zpráv. Když adresát očekává podepsanou zprávu a místo ní přijde zpráva holá, může ho to upozornit na potenciální riziko. Pokud chce odesílatel zajistit, že odchycením zprávy nemůže dojít k jejímu přečtení, zašifruje ji klíčem adresáta.

Nejčastější e-mailové klienty podporu PGP mají – přímo nebo s pomocí nějakého rozšíření. Cílem této práce je vytvoření řešení v jazyce Python, které budou moci využívat další aplikace.

Ke květnu 2021 jsou pro produkční nasazení podporovány čtyři verze jazyka: 3.6 až 3.9. Některé aplikace podporují i historické verze (například `python-gnupg`, kterou jsem využil při implementaci své knihovny `python-gnupg-mail`, garantuje podporu i pro Python 2.7) [3, 4, 5].

Výsledkem praktické části bude program využitelný samostatně i jako knihovna. Výsledkem zpracování souboru bude sada objektů popisujících části e-mailu s přidanými atributy, především informacemi o ověření PGP.

1 PGP

PGP (Pretty Good Privacy, *Celkem dobré soukromí*) je rodina kryptografických systémů, kterou vytvořil Philip R. Zimmermann na počátku devadesátých let. Jeho cílem bylo definování univerzálního a bezpečného sdílení informací přes internet. V dokumentaci k první verzi uvádí:

> *Perhaps you think your E-mail is legitimate enough that encryption is unwarranted. If you really are a law-abiding citizen with nothing to hide, then why don't you always send your paper mail on postcards? Why not submit to drug testing on demand? Why require a warrant for police searches of your house? Are you trying to hide something? You must be a subversive or a drug dealer if you hide your mail inside envelopes. Or maybe a paranoid nut. Do law-abiding citizens have any need to encrypt their E-mail?* [6, str. 27]

V době vytvoření PGP byl dle americké legislativy export bezpečných kryptosystémů mimo USA vázán stejnými zákony jako vývoz zbraní a Zimmermannovi hrozily sankce. Historické okolnosti vzniku jsou shrnuty v první části této kapitoly.

OpenPGP je protokol postavený na PGP ve verzi 5.x. Vznikl v roce 1997 kvůli nejistotě právních vztahů ohledně patentů v rámci firmy PGP Inc. V této souvislosti uvnitř IETF (*Internet Engineering Task Force*) vznikla skupina OpenPGP Working Group, která správu standardu dostala na starost. Struktura zpráv OpenPGP dle nejnovějšího RFC (*Request for Comments*) je popsána v druhé podkapitole.

Ve třetí části jsou shrnuty problémy PGP: nejčastější výtky a kritické názory, které se během let objevily.

V poslední části je zmíněno GPG (nebo také GnuPG, GNU Privacy Guard), což je otevřená implementace OpenPGP od Free Software Foundation, která kompletně odpovídá standardu IETF.

1.1 Vývoj PGP

První verzi PGP začal Phil Zimmermann psát na začátku devadesátých let jako nástroj pro lidskoprávní organizace a uskupení. *„You don't take a human rights tool and put backdoor in it,“* řekl na konferenci DEFCON v roce 2003 [7].

Z části šlo také o reakci na Senátní návrh zákona č. 266 z roku 1991, který měl vynucovat backdoor ve veškerém softwaru umožňujícím šifrování s velikostí klíčů nad 40 bitů. Ke schválení nakonec nedošlo. PGP 1.0 využívalo asymetrické šifrování RSA v kombinaci s vlastní symetrickou šifrou BassOmatic (později nahrazenou šifrou

IDEA kvůli své slabosti), hashovací funkcí MD4 a algoritmem *uuencode* pro konverzi na sedmibitový obsah [6, 8].

Známí Phila Zimmermanna kód nahráli na americké bulletin boardy, ze kterých se PGP rozšířilo mimo hranice do světa. MIT poté Zimmermannovi zaslalo upozornění, že porušuje jejich patent pro RSA; tento spor byl později vyřešen. V roce 1993 Zimmermanna začali vyšetřovat kvůli vývozu kryptografických nástrojů mimo USA, což bylo dle tehdejších zákonů nelegální: pod kategorii *munitions export without a licence* spadala i distribuce kryptografických algoritmů s velikostí klíče nad 40 bitů [8].

PGP 2 (1992) využilo šifry RSA, IDEA, hash MD5 a funkci radix-64. Zimmermann spolupracoval s programátory a výzkumníky ze zahraničí, a to bylo pro jeho probíhající soud problematické, protože šlo o ilegální aktivitu vývoje kryptografie mimo USA [7, 9].

V roce 1993 dostal Phil Karn, aktivní přispěvatel IETF, povolení distribuovat svůj kryptografický algoritmus vytištěný v knize, ale ne v digitální podobě. Zimmermann se na základě této informace rozhodl kód PGP vydat ve formě knihy – publikace o 934 stranách nesla název *PGP: Source Code and Internals*. Matematik Dan Bernstein ve spolupráci s EFF (*Electronic Frontier Foundation*) podal kvůli zákazu šíření kryptografických algoritmů mimo USA žalobu proti Ministerstvu spravedlnosti Spojených států amerických. Tyto tři události jsou součástí tzv. *Crypto Wars* a podílely se na povolení exportu kryptografie z USA do světa. K uvolnění pravidel a zrušení soudních sporů došlo v roce 2000 [8, 10, 11].

Zimmermann založil PGP Inc., která během dvou let přešla pod společnost Viacrypt. Společná dohoda stanovovala, že Viacrypto bude vydávat sudé verze PGP a PGP Inc. liché. Viacrypt svou verzi vydala (dříve než on) jako PGP 4, a aby nedocházelo k nepochopení a komplikacím, následující Zimmermannova verze byla pátá (třetí byla zcela tedy přeskočena) [12].

Verze 5 (1997) opravila bezpečnostní chyby verze předchozí a přidala algoritmy DSA a ElGamal, které nebyly patentově zatížené. Došlo také ke změně struktury softwaru: místo programu pro příkazovou řádku bylo PGP 5 použitelné také jako knihovna.

V tomto roce práva na PGP odkoupila firma Network Associates, Inc. a po změně zákona o exportu v roce 2000 přestala zveřejňovat zdrojové kódy [13]. Během pěti let práva přešla na firmu bývalých zaměstnanců PGP, nazvanou PGP Corporation, která byla později koupena firmou Symantec.

1.2 OpenPGP

OpenPGP je protokol založený na PGP 5, poslední open-source verzi Pretty Good Privacy. Definuje formát pro šifrování a podepisování zpráv, pro výměnu veřejných klíčů či způsoby pro vytváření jejich vzájemné důvěry.

V rámci IETF byla v roce 1997 vytvořena OpenPGP Working Group za účelem vytvoření otevřeného standardu, který se ve formě proprietárního software používal od roku 1991. O rok později vydala dokument OpenPGP Message Format (RFC 2440), který se stal referencí pro téměř všechny implementace systému PGP v rámci internetu [14].

Hlavním cílem OpenPGP je zajištění integrity přenášených zpráv pomocí využití digitálních podpisů, šifrování, komprese a konverze do radix-64. Kombinuje symetrickou a asymetrickou kryptografii pro zajištění maximální důvěrnosti. Aktuální verze standardu byla vydána v roce 2007 v RFC 4880 [12].

1.2.1 Kryptografický princip OpenPGP

Zpráva je šifrována symetrickým algoritmem s využitím jednorázového klíče (*session key*), který je zašifrován veřejným klíčem adresáta a přenášen spolu se zprávou.

Digitální podpis a zašifrování zprávy je možné zkombinovat: nejprve je vygenerován podpis, který je přiložen ke zprávě, potom je celá zpráva i s podpisem zašifrována symetrickým relačním klíčem. Tento klíč je zašifrován veřejným klíčem příjemce a v zašifrované formě je umístěn před zašifrovaný blok dat.

Šifrované zprávy, klíče i podpisy jsou řetězce libovolných oktetů (bajtů). Některé systémy podporují pouze tisknutelné sedmibitové znaky. Z toho důvodu je definována oboustraná konverze do formátu radix-64, nazývaného také *ASCII Armor*, která přenos libovolných dat po sedmibitovém médiu umožňuje.

Veřejné klíče jsou uloženy v systémové klíčence, což může být jeden či více souborů nebo databáze.

Protokol pracuje s tzv. S2K specifikátory (*String-to-Key Specifiers*), které převádí heslo na symetrické klíče pro (de)šifrování. Jsou využity při zašifrování tajné části soukromého klíče v klíčence a během konverze hesla při šifrování klíčů pro symetricky zašifrované zprávy. Tyto specifikátory mohou být jednoduché (vstupní řetězec je přímo hashován), solené (před vstupní řetězec je přidáno osm oktetů soli) a iterované solené (osolený vstup je hashován vícekrát). Konkrétní postup záleží na užité šifře a velikosti výstupu hashovací funkce.

1.2.2 OpenPGP pakety

Každá zpráva je složena z více záznamů nazývaných pakety, které jsou buď vnořeny do sebe, nebo poskládány za sebou.

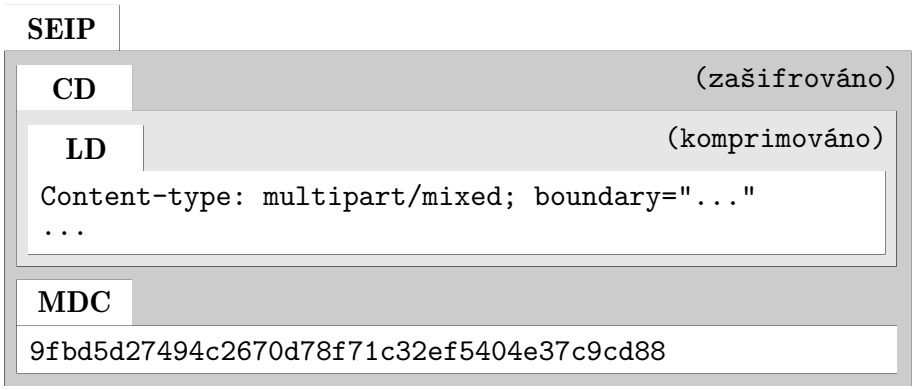
OpenPGP paket se skládá z tagu (informace o typu paketu), délce v bajtech a těla, v němž jsou obsažena samotná data. RFC 4880 definuje osmnáct hlavních tagů: veřejné a soukromé klíče, symetricky zašifrovaná data, zkomprimovaná data a podobně [12].

Signature Packet obsahuje verzi signatury, typ podpisu, algoritmy klíčů a hashů, informace o subpaketech či samotný podpis. Podepisována je kompletní hodnota hashe počítaného od verze signatury po subpacketová data. Dva oktety hashe jsou přidány do hlavičky pro možnost rychlejšího zamítnutí poškozených podpisů. Subpakety nesou informace o paketu: časové značky, revokabilitu nebo preferované algoritmy.

Compressed Data Packet většinou obaluje paket obsahující přenášená data (*Literal Data Packet*). Data jsou komprimovaná pomocí *deflate*, a to algoritmy ZIP, ZLIB nebo BZIP2.

Symmetrically Encrypted Data Packet obsahuje data zašifrovaná symetrickým algoritmem; po dešifrování obsahuje další paket(y), které chrání. Pro šifrování je použita bloková šifra v CFB módu (viz kapitolu 1.2.5).

Literal Data Packet obsahuje informaci o kódování (`0x62` binární, `0x74` text, `0x75` UTF-8 text), název souboru a text samotný. Řádky jsou odděleny pomocí bajtů `CRLF` [12].



Obr. 1.1: Vrstvení PGP paketů. SEIP (*Symetrically Encrypted and Integrity Protected*), CD (*Compressed Data*), LD (*Literal Data*), MDC (*Modification Detection*). Převzato z [39, str. 554].

1.2.3 ASCII Armor

Pro reprezentaci dat OpenPGP využívá proud libovolných oktetů. Některé systémy mohou s daty manipulovat při transportu nebo ukládání, přenášený obsah je zároveň citlivý na změny. OpenPGP definuje jednotný způsob konverze oktetů do tisknutelných znaků, zápis takových dat (po převedení do radix-64) se nazývá ASCII Armor.

Radix-64 se skládá ze dvou částí: binárních dat (kódovaných v base64) a kontrolního součtu (CRC-24 převedený do base64 umístěný pod zprávou na samostatném řádku, který začíná znakem `=`).

ASCII Armor data jsou složena z hlavičky, samotného ASCII obsahu, kontrolního součtu a patičky. Hlavička je složena z pěti ASCII znaků pro minus na začátku i konci řádku, mezi nimi je řetězec znaků informující o typu obsahu (přehled těchto řetězců je v tabulce 1.1). Tato data musí začínat na novém řádku a nový řádek po nich musí ihned následovat.

Tab. 1.1: Hlavičky ASCII Armor bloku

`BEGIN PGP MESSAGE`	Zašifrované, podepsané, zkomprimované soubory
`BEGIN PGP PUBLIC BLOCK KEY`	Veřejný klíč
`BEGIN PGP PRIVATE KEY BLOCK`	Privátní klíč
`BEGIN PGP MESSAGE, PART X/Y`	X-tá zpráva z Y
`BEGIN PGP MESSAGE, X`	X-tá zpráva z neznámého počtu
`BEGIN PGP SIGNATURE`	Oddělený podpis či podpis OpenPGP/MIME

Hlavičky se zapisují ve formátu `název: hodnota` a nejsou součástí podepisované části. Definovanými klíči pro hlavičky jsou `Version`, `Comment`, `MessageID` (identifikátor zprávy v případě více částí), `Hash` a `Charset` (znaková sada, pokud jde o jiné kódování než UTF-8).

Patička je shodná s hlavičkou, jen je klíčové slovo `BEGIN` nahrazeno slovem `END`.

Samotné kódování base64 spočívá v rozdělení proudu oktetů na proud znaků po šesti bitech, které jsou kódovány jako 64 tisknutelných znaků (`A-Za-z0-9+/`) do řádků po 76 znacích. Pokud na konci zprávy zůstanou čtyři bity, jsou za ně vloženy dva bity nulové a za zakódovanou část je vložen znak `=`. Pokud zpráva končí dvěma bity, jsou vloženy nulové bity čtyři a za konec jsou vloženy znaky `==`.

Ukázka *armored* zprávy je ve výpisu 1.1.

1.2.4 Cleartext Signature

Otevřený podpis ponechává podepisovaný text čitelný i bez využití speciálního programu (nedochází ke konverzi do jiné reprezentace). Zpráva podepsaná tímto způ-

```
-----BEGIN PGP MESSAGE-----
Version: OpenPrivacy 0.99

T3ByYXZkb3bDoSBQR1AgZGF0YSBieWxhIG1vYyBkbG91aMOhLCBOZW50byBOZ
XhOIGplIGtyYXTFocOtIG7DoWhyYWRhLgo=
=njUN
-----END PGP MESSAGE-----
```

Výpis 1.1: Ukázka ASCII Armored zprávy (Zdroj: [12, str. 59])

sobem začíná řádkem obsahujícím text `BEGIN PGP SIGNED MESSAGE` a je tvořena hlavičkami, textem zprávy a podpisem s hlavičkou `BEGIN PGP SIGNATURE`.

Aby mohl parser rozlišit PGP data a podepisovaný text, před všechny řádky začínající znakem minus je jsou vloženy znaky minus a mezera („- “). Tento princip je nazván *Dash-Escaping*. Podpis je počítán z textu s `CRLF` konci řádků. Všechny *whitespace* znaky (mezery a tabulátory) na konci řádku musí být ignorovány [12].

1.2.5 Algoritmy využívané v OpenPGP

Každá PGP zpráva obsahuje oktet, který jedoznačně určuje algoritmus využitý k zakódování zprávy.

RFC 4880 podporu pro část algoritmů vynucuje, pro část doporučuje a pro některé jsou jen vynechaná místa pro budoucí použití. Pro asymetrickou kryptografii jsou zmíněny RSA, ElGamal, DSA s rezervacemi pro EC a ECDSA, pro symetrickou IDEA, TripleDES, Blowfish nebo AES.

Povolené komprimační algoritmy jsou ZIP, ZLIB a BZIP2. Povolené hashovací algoritmy jsou MD5, SHA-1, SHA-256, SHA-384, SHA-512, SHA-224 s tím, že algoritmus MD5 je prohlášen za zastaralý a je s ním zakázáno vytvářet nové podpisy.

Kompletní seznam algoritmů včetně oktetů k nim přiřazených popisuje RFC 4880 v sekci *Constants* [12, str. 61].

1.2.6 Otisky veřejných PGP klíčů

Fingerprint (otisk) je způsob kratšího vyjádření dat, v tomto případě veřejného klíče. Celý paket obsahující veřejný klíč je spolu s oktetem `0x99` a délkou paketu zahashován funkcí SHA-1, čímž je získáno číslo o velikosti 160 bitů.

Toto číslo je převedeno na hexadecimální tvar a typicky bývá rozděleno na deset bloků po čtyřech znacích. Takovou reprezentaci je možné vytisknout na vizitku nebo jiné fyzické médium a umožnit tak ověření veřejného klíče bez nutnosti ho celý přepsat do počítače.

```
C70D 892B FF44 30A2 9546 0EE3 FC34 5691 4E63 4265
95C5 579C 0B5C 08E8 791C 7C6A 5646 C49C 772E DD30
```

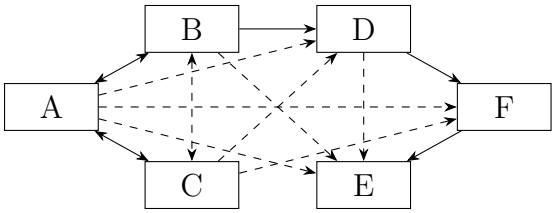
Výpis 1.2: Dvě ukázky PGP otisků

Použití hashovací funkce znamená větší šanci, že dva různé veřejné klíče budou reprezentovány totožně. SHA-1 je ale považována za bezpečnou, umělé vytvoření kolize není při současném výkonu počítačů pokládáno za možné a nejedná se o snížení bezpečnosti této funkce.

1.2.7 Síť důvěry a keyservery

Princip sítě důvěry vychází z předpokladu, že veřejný klíč stažený z internetu nenese žádnou záruku o věrohodnosti a vazbě zadaného jména a e-mailu k reálné identitě.

PGP nabízí možnost podepsat cizí veřejný klíč a tím ho „autorizovat“ svým jménem. Vzájemně podepsané veřejné klíče tvoří Síť důvěry (*The Web of Trust*), díky které si mohou důvěřovat dva zcela nezávislé klíče – za předpokladu, že mezi nimi existuje jeden či více klíčů, kterým důvěřují [9].



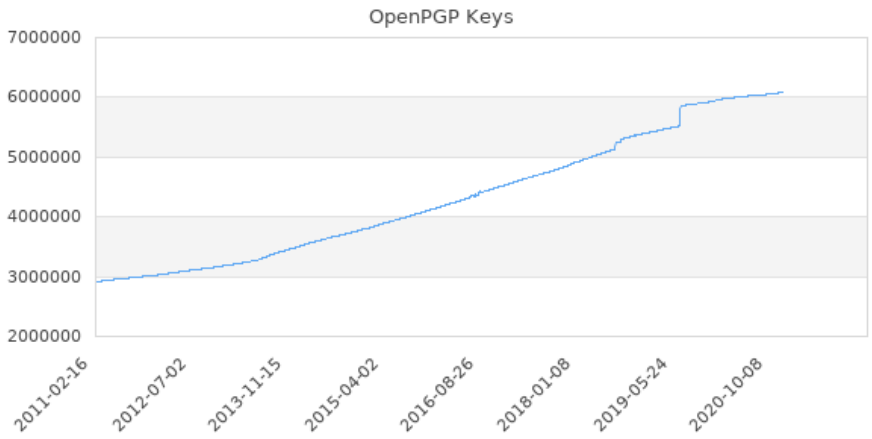
Obr. 1.2: Síť důvěry. Plné čáry popisují přímou důvěru a čárkované nepřímou, šipky znázorňují směr důvěry (od věřícího k důvěryhodnému). Účastník A věří veřejnému klíči účastníka E, protože (přímo) věří účastníkovi B, který (pro A již nepřímo) věří účastníkovi D, ten E a ten F. Zdroj: Vlastní zpracování dle [15].

Keyserver je volně přístupný server držící veřejné PGP klíče. Jakýkoli uživatel může na takový server svůj klíč nahrát a tím ho zpřístupnit ostatním. Tyto keyservery ale trpí problémy vázanými především na ochranu soukromí – je velmi jednoduché z klíčů získávat e-mailové adresy a ty využívat například ke spamu. Je také prakticky nemožné jednou nahraný klíč smazat, jedinou možností bývá nahrání revokačního certifikátu, čímž dojde k deaktivaci klíče.

1.3 Limitace OpenPGP a jeho použití v praxi

Během let rostl počet hlasů z řad akademiků i techniků, kteří doporučovali systém PGP nepoužívat. Zvláště v posledních letech, kdy se začal v mnohých mobilních aplikacích využívat Signal protokol, se objevily blogové příspěvky na stránkách současných oborníků a profesionálů zabývajících se kryptografií.

Jedním z problémů je malé využití PGP. Za celou dobu existence bylo na SKS keyserver nahráno šest milionů PGP klíčů, jak ukazuje obrázek 1.3. Tento fakt na svém blogu komentuje Moxie Marlinspike, jeden z autorů Double Ratchet algoritmu: *„By today's standards, that's a shockingly small user base for a month of activity, much less 20 years,"* [16].



Obr. 1.3: Počet veřejných klíčů na SKS keyserveru (Zdroj: [17])

Dalším problémem je absence dopředné bezpečnosti (*forward secrecy*): při uniknutí soukromého klíče lze dešifrovat veškerou minulou i budoucí komunikaci. Daný problém řeší z moderních systémů například Signal protokol od Open Whisper Systems. Pro své fungování používá algoritmus Double Ratchet, který mění šifrovací klíč s každou zprávou [18].

PGP je složitý systém. *„Instead of developing opinionated software with a simple interface, GPG was written to be as powerful and flexible as possible. It's up to the user whether the underlying cipher is SERPENT or IDEA or TwoFish,"* píše Marlinspike [16]. Pro člověka, který není seznámen s kryptografickými principy, je nastavení komplikované a matoucí.

Kryptograf Matthew Green na svém blogu uvádí další důvody: keyservery jsou nepřehledné a zvyšují komplexitu, budování sítě důvěry je problematické a může narušit anonymitu, na text je aplikována komprese, PGP udržuje zpětnou kompatibilitu s kryptografií devadesátých let [19].

1.4 GnuPG

GNU Privacy Guard je otevřená implementace RFC 4880. Mimo OpenPGP podporuje také S/MIME a integraci s ssh a HKP keyservery [20].

Projekt byl v roce 1997 založen Wernerem Kochem, v tu chvíli ještě nezávisle na OpenPGP. Verze 2.0 z roku 2006 byla interně nazývána NewPG, protože šlo o velkou změnu struktury kódu, která přidala i podporu S/MIME. Je vyvíjen firmou g10 Code, kterou Koch založil.

Jde o jedinou oficiální, plně funkční implementaci OpenPGP, proto ho využívá i Python skrz knihovnu `python-gnupg` (viz kapitolu 4.1.1).

2 E-mail

S postupným rozšiřováním internetu rostly možnosti digitální komunikace. Zprávy se elektronicky vyměňovaly téměř od počátku existence tehdejšího internetu (první dokumenty ještě užívají termín ARPANET), bylo ale nutné vytvořit univerzální způsob jejich automatického zasílání.

Tato kapitola popisuje e-mail od jeho vzniku na počátku sedmdesátých let po současnou sadu RFC 2045 až 2049 a RFC 5322.

2.1 Network mail

První dokument snažící se sjednotit formátování elektronické pošty, které by se již podobalo systému používanému v současnosti, bylo RFC 561 (Standardizing Network Mail Headers):

> *One of the deficiences of the current FTP mail protocol is that it makes no provision for the explicit specification of such header information as author, title, and date. Many systems send that information, but each in a different format. One fairly serious result of this lack of standardization is that it's next to impossible for a system or user program to intelligently process incoming mail* [21].

Dokument navrhl použití hlaviček identifikujících jednotlivé položky: `From`, `Date`, `Subject` a naznačil položku `Misc`, kterou však dále nerozvedl. Po hlavičkách následoval prázdný řádek a další řádky se považovaly za obsah zprávy. Text e-mailu mohl obsahovat jakékoliv znaky základní ASCII tabulky krom `CR` a `LF` (*Carriage Return*, *Line Feed*). U názvů položek se nehledí na kapitalizaci písmen: `From`, `FROM` i `FRom` jsou si rovnocenné.

```
1 Date:    18 OCT 2020 1852-GMT
2 From:    xhorky31 AT vutbr
3 To:      xfiter00 AT vutbr
4 Subject: Ukazka zpravy dle RFC 581
5
6 V e-mailu jde pouzit pouze znaky zakladni ASCII tabulky. RFC
7 neklade zadne  naroky na delku radku,  ukazka v dokumentu je
8 zalomena na padesatem znaku. Kazdy radek konci znaky <CRLF>.
```

Výpis 2.1: Ukázka e-mailu dle RFC 581

O dva roky později byl původní dokument rozšířen vydáním RFC 680 (Message Transmission Protocol). Jeho primárním cílem bylo definování nových hlaviček; autoři také požadovali, aby obsah mohl přečíst jak člověk, tak stroj [22].

Vedle položky `From`, která označuje iniciátora zprávy, je položka `Sender`, která definuje odesílatele. Poprvé se objevuje `Message-ID` sloužící k jednoznačné identifikaci zprávy, `To` (adresáti), `Cc` (sekundární adresáti) a `Bcc` (adresáti skrytí prvním dvěma skupinám).

Další položky, které dnes již využívány nejsou, byly například `Authentication` (ověření odesílatele), `References` (odkazy na předešlé zprávy), `Keywords` (klíčová slova) nebo `Precedence` (priorita zprávy).

2.1.1 RFC 733

RFC 733 (Standard for the Format of Text Messages) v roce 1977 zpřesnilo možné hodnoty hlaviček. Uživatel může mít více poštovních schránek pod různými jmény na více zařízeních. Zároveň definovalo skupiny příjemců a povolilo i adresy mimo ARPANET (například fyzickou poštu).

Návrh tohoto dokumentu uvádí: *„No mechanism for authentication is provided, since the Network provides no mechanisms for enforcing mail security,“* [23, str. 5].

Bylo navrženo dodnes používané zalamování hlaviček:

```
To: "Matyáš Horký"
  <xhorky31 at vutbr.cz>
```

je funkčním ekvivalentem

```
To: "Matyáš Horký" <xhorky31 at vutbr.cz>
```

za předpokladu, že je na novém řádku alespoň jeden znak mezery nebo tabulátoru (tzv. *whitespace*). Těla jednotlivých hlaviček mohou obsahovat jakékoliv tisknutelné a whitespace znaky kromě dvojtečky.

Pro délku řádku limit nebyl stanoven, je pouze zmíněno doporučení nepoužívat řádky delší než 65–72 znaků. Současně není doporučeno používat tabulátory kvůli neshodě na jeho reprezentaci (délce) napříč systémy [24].

V tabulce 2.1 je přehled vybraných hlaviček. RFC 733 jich definuje více, pouze některé jsou z hlediska této práce důležité.

2.1.2 RFC 822

Následující RFC 822 používání hlaviček zpřísnilo a přizpůsobilo rostoucímu ARPANETu několika způsoby: dovolilo použití speciálních znaků v obsahu textu pomocí

Tab. 2.1: Vybrané hlavičky dle RFC 733

`From`	Identita uživatelského účtu
`Sender`	Identita odesílatele (možno vynechat, pokud je stejná s položkou `From`)
`Reply-To`	Adresy, na které by měla být zaslána odpověď
`To`	Identity primárních příjemců
`Cc`	Identity sekundárních příjemců
`Bcc`	Skrytí příjemci (vzájemně se vidět mohou i nemusí, dle implementace)
`Subject`	Předmět zprávy, sumarizace obsahu
`Message-ID`	Unikátní identifikátor zprávy (v rámci stroje)
`Date`	Časová značka

zpětného lomítka a zavedlo hlavičky přidávané každým mail serverem (`Received` s hodnotami `From`, `By`, `Via`, ...) [25].

K tomu přibyla podpora pro přeposílání zpráv (zdvojení některých hlaviček s prefixem `Resent-`) a pro neoficiální rozšíření hlaviček (s prefixem `X-`). Místo slovní značky `at` byla stanovena povinnost používat zavináč.

V části 4.7.3 je zmíněno šifrování s parametrem `Encryption`:

> *Sometimes, data encryption is used to increase the privacy of message contents. If the body of a message has been encrypted, to keep its contents private, the "Encrypted" field can be used to note the fact and to indicate the nature of the encryption. The first <word> parameter indicates the software used to encrypt the body, and the second, optional <word> is intended to aid the recipient in selecting the proper decryption key. This code word may be viewed as an index to a table of keys held by the recipient.*
>
> *Note: Unfortunately, headers must contain envelope, as well as contents, information. Consequently, it is necessary that they remain unencrypted, so that mail transport services may access them. Since names, addresses, and "Subject" field contents may contain sensitive information, this requirement limits total message privacy* [25, str. 24].

RFC 822 kromě konkretizace obsahu polí definovalo také e-mailovou adresu, kterou systém musel vždy přijmout: `postmaster@domain`, a to bez ohledu na kapitalizaci. *„The local-part 'Postmaster' has been reserved, so that users can be guaranteed at least one valid address at a site,"* [25, str. 43].

```
Date:    21 October 2020 0154+0200
From:    Matyas Horky (Komentar uvnitr zavorek se neodesila
         a slouzi pouze k lokalni identifikaci)
         <xhorky31@vutbr>
Sender:  Ivana Fitere <xfiter00@vutbr>
To:      admin at vutbr
Bcc:     <xfiter00@vutbr>
Subject: Ukazka zpravy dle RFC 733
Message-ID: <2020.10.18-18.52.04-034@vutbr>

Stale lze pouzivat pouze zakladni ASCII znaky. Radky konci
CRLF. Po prvnim prazdnem radku se vse bere jako obsah zpravy.
```

Výpis 2.2: Ukázka e-mailu dle RFC 733

2.2 Internet Message Format

Nejnovějším platným standardem je RFC 5322 (Internet Message Format) z roku 2008, které bylo publikováno společně s novou definicí protokolu SMTP (RFC 5321). Vychází z dokumentů zmíněných v předchozích kapitolách (RFC 822–2822) a dokumentů MIME (viz kapitolu 2.3) [26].

V úvodní části textu RFC 5322 stanovuje, že znaky přenášené zprávy musí být ve znakové sadě US-ASCII, tj. v hodnotě 1 až 127. Pro neanglický text tento fakt znamená nutnost kódovat diakritiku, všech jazyků se týká problematika binárních dat obsažených ve zprávě jako přílohy. Obojí řeší právě MIME (viz kapitolu 2.3).

Obecná pravidla hlaviček dle RFC 5322:

- Řádek je řetězec znaků, který je ukončený znaky `CRLF`,
- zpráva je složena z hlaviček a nepovinně těla,
- tělo je sekvence znaků pod hlavičkou, od které je odděleno prázdným řádkem,
- řádek nesmí být delší než 998 znaků, neměl by přesahovat 78 znaků,
- název hlavičky je od těla oddělen dvojtečkou,
- hlavičky lze skládat: každé tělo může být rozděleno ve *whitespace* místě tak, že jsou před něj vloženy znaky `CRLF`, další řádek začíná alespoň jedním whitespace znakem,
- rozložená hlavička není počtem znaků omezena.

Jediné povinné hlavičky jsou datum a odesílatel. Pro obsah platí stejná pravidla pro délku řádků a výskyt bajtů `CRLF` jako pro hlavičky.

RFC 5322 také spoustu pravidel definovaných v minulosti prohlašuje za zastaralou se záměrem zjednodušit parsování zprávy, vynucuje formátování časové zóny

```
Date:    23 October 2020 0050+0200
From:    Matyas Horky <xhorky31@vutbr.cz>
Sender:  Ivana Fitere <xfiter00@vutbr.cz>
To:      <postmaster@vutbr>
Bcc:     <xfiter00@vutbr>
Subject: Ukazka zpravy dle RFC 822
Comment: Komentar, ktery neni soucasti samotne zpravy
Message-ID: <2020.08.23.00.50.13-376@vutbr.cz>

Pravidla  pro  obsah  poli jsou  prisnejsi, jiz se blizi
dnesnim pozadavkum.  Stale vsak neni  nutne specifikovat
celou adresu az ke korenove  domene, pokud jde o interni
komunikaci.
```

Výpis 2.3: Ukázka e-mailu dle RFC 822

jako čtyřmístné číslo se znaménkem a zakazuje rozložení řádku mezi názvem hlavičky a dvojtečkou.

Součástí hlaviček jsou také *Trace fields* přidávané SMTP servery, jejich definice a pravidla jsou obsažena v RFC 5321 [27]. Popis fungování distribuce zpráv je popsán v kapitole 2.6.

2.3 MIME

Multipurpose Internet Mail Extensions jsou RFC rozšiřující jednoduchou definici formátu elektronické pošty dle RFC 822. Jsou psána s ohledem na kompatibilitu s RFC 822: pravidla těla zpráv se od doby jejich vydání nezměnila.

Koncept MIME byl poprvé představen roku 1992 v RFC 1341. Nový systém musel být stoprocentně funkční v historických implementacích:

> *Several of the mechanisms described in this document may seem somewhat strange or even baroque at first reading. It is important to note that compatibility with existing standards AND robustness across existing practice were two of the highest priorities of the working group that developed this document. In particular, compatibility was always favored over elegance* [28, str. 3].

Tab. 2.2: Vybrané hlavičky dle RFC 5322

`From`	Identita uživatelského účtu (uživatelských účtů)
`Sender`	Identita odesílatele (možno vynechat, pokud je stejná s položkou `From`)
`Reply-To`	Adresy na které by měla být zaslána odpověď
`To`	Identity primárních příjemců
`Cc`	Identity sekundárních příjemců
`Bcc`	Skrytí příjemci (vzájemně se vidět mohou i nemusí v závislosti na implementaci; viz [26, kap. 5])
`Subject`	Předmět zprávy, sumarizace obsahu
`Comments`	Komentáře ke zprávě
`Keywords`	Klíčová slova
`Message-ID`	Unikátní identifikátor zprávy (v rámci domény)
`In-Reply-To`	Seznam identifikátorů zpráv, na které zpráva odpovídá
`References`	Odkazy na předchozí vlákno konverzace
`Date`	Časová značka odeslání zprávy

2.3.1 MIME hlavičky

Aby byla zpráva rozpoznána jako MIME, musí obsahovat hlavičku `MIME-Version`. Každé pole musí mít svou hlavičku `Content-Type`, jejíž hodnoty jsou popsány v tabulce 2.3. Pokud je tato hlavička vynechána, interpretující systém by se k poli měl chovat tak, jako by měl obsah formátu `text/plain; charset=US-ASCII`.

Protože SMTP protokol (dle RFC 821) přenáší sedmibitová ASCII data, bylo nutné vytvořit mechanismy pro konverzi obecných dat a to signalizační hlavičkou `Content-Transfer-Encoding`. Dovoluje použití hodnot `base64`, `quoted-printable`, `8bit`, `7bit`, `binary` a kódování mimo standard s hlavičkou začínající `x-`. Osmibitový a binární přenos dokument povoluje pro typy přenosu jiným způsobem, než je protokol SMTP, který vyžaduje sedmibitový text.

`Quoted-Printable` obsah je navržen pro téměř-US-ASCII data (například český text). Znaky bez diakritiky se nezmění, znaky s ní (s decimální hodnotou nad 128) jsou uloženy jako `=` následované dvěma hexadecimálními znaky reprezentujícími jeden bajt. Znak Š bude zapsán jako `=A9`, ý jako `=FD` atd. Řádky delší než 76 znaků se musí rozdělit rovnítkem ihned následovaným znaky `CRLF`.

2.3.2 Pole MIME zprávy

Jednotlivá pole jsou definována pomocí `boundary`. Hranice pole začíná dvěma minus znaky (`--`), následují znaky ohraničení (ty musí být z tisknutelné části znakové sady

```
Date:    Sat, 24 Oct 2020 15:33:17+0200
From:    Matyas Horky <xhorky31@vutbr.cz>
Sender:  Ivana Fitere <xfiter00@vutbr.cz>
To:      postmaster@vutbr.cz
Bcc:     xfiter00@vutbr.cz
Subject: Ukazka zpravy dle RFC 5322
Comment: Komentar, ktery neni soucasti samotne zpravy
Message-ID: <2020.08.24.15.33.17-679@vutbr.cz>

Text e-mailu. Plati stejna pravidla jako drive: nelze posilat
zpravy, ktere maji znakovou sadu mimo US-ASCII.

Pro podporu diakritiky, priloh ci sifrovani je treba zpravu
odeslat jako MIME.
```

Výpis 2.4: Ukázka e-mailu dle RFC 5322

```
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: base64
```

Výpis 2.5: Ukázka `Content-Type` hlavičky dle RFC 1341

US-ASCII) a nový řádek. Tento systém vychází z RFC 934 (Proposed Standard for Message Encapsulation). Ohraničení se nesmí vyskytovat v textu těla zprávy. V praxi se tento požadavek řeší vygenerováním dostatečně dlouhé náhodné sekvence znaků [28, 29].

Pole může být ukončeno dvěma způsoby:

- otevírací značkou (předchozí pole je automaticky ukončeno),
- uzavírací značkou, která odpovídá značce otevírací s `--` na konci – to v případě, že již žádné další pole v dané MIME skupině nenásleduje.

V roce 1993 vyšla malá revize RFC 1341 s číslem 1521, která z hlediska této bakalářské práce nepřinesla nic nového. Společně bylo ale vydáno RFC 1522, které v hlavičkách umožnilo používání libovolných znaků, ne jen základní sadu US-ASCII. Zakódovaná data se zabalují do značek `=?` a `?=`, vnitřní pole jsou oddělena otazníky. Na první pozici je znaková sada (na ukázce 2.7 Latin 2 pro středoevropské rozšíření ASCII) a na druhé typ kódování (`Q` pro `quoted-printable`, `B` pro `base64`). Tento systém se využívá dodnes.

V roce 1996 vyšla poslední revize MIME formátu: RFC 2045 až 2049. Pravidla polí zmíněna výše zůstala zachována; dokumenty se zaměřují na opravy chyb.

Tab. 2.3: Kategorie `Content-Type` dle RFC 1341

`text`	Text. Podtyp `plain` indikuje neformátovaný text; ostatní podtypy mohou vylepšit vzhled textu, jako například `richtext`.
`multipart`	Zapouzdření více typů: `mixed` jako základní typ, `alternative` pro stejný obsah v jiných formátech, `parallel` pro části zpráv určené k náhledu v jednu chvíli, `digest` pro více zpráv dle RFC 822.
`message`	Zpráva. Tělo základnícho podtypu `rfc822` je plně formátovaná zpráva dle RFC 822, `partial` označuje zprávu nezaslanou kompletně (fragmentace), `external-body` odkazuje na velká data mimo zprávu.
`image`	Obrázek. Podtypy jsou `jpeg` a `gif`.
`audio`	Zvuk. Jediný specifikovaný podtyp je `basic`.
`video`	Video. Jediný specifikovaný podtyp je `mpeg`.
`application`	Jiný typ dat. `Octet-stream` pro neinterpretovaná binární data, `ODA` a `PostScript`.

2.4 MIME a PGP

RFC 1847 definovalo dvě další `Content-Type` hlavičky, jmenovitě `multipart/signed` a `multipart/encrypted`. Nedefinuje bezpečnostní ochranu, pouze popisuje propojení MIME s kryptografickými systémy, které e-mailový klient může využít [30].

RFC 2015 integruje PGP do MIME. Využívá k tomu návrh RFC 1848 (MIME Object Security Services), ze kterého si půjčuje strukturu přenosu dat asymetrickou kryptografií [31]. Jeho aktualizace, RFC 3156 z roku 2001, je nejnovější verzí specifikace zabývající se integrací PGP do e-mailu. Definuje tři hlavičky obsahu:

- `application/pgp-encrypted` pro šifrovaný obsah,
- `application/pgp-signature` pro PGP podpis,
- `application/pgp-keys` pro PGP klíče [32].

Podpis je počítán z kompletní zprávy včetně definice obsahu. Ignoruje tedy MIME hranici, ale `Content-Type` hlavičku už ne. Ukázka takové zprávy je ve výpisu 2.8; podepsaná část jsou řádky 7 až 19 včetně.

E-mail s obsahem `multipart/encrypted` se musí skládat právě ze dvou částí: pole `pgp-encrypted` (obsah musí být `Version 1`) a `application/octet-stream` (obsahuje samotnou zašifrovanou zprávu).

E-mail s obsahem `multipart/signed` se musí skládat právě ze dvou částí: samotného MIME obsahu a pole `application/pgp-signature`, které obsahuje *detached* PGP podpis.

Pro kombinaci obou předchozích funkcí PGP lze využít dva způsoby: první je za-

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=MAIL-BOUNDARY

Uvodni hlavicka. Mela by byt ignorovana, muze vsak obsahovat
informaci pro e-mailove klienty, kteri multipart zpravy
nepodporuji.
--MAIL-BOUNDARY

Tato cast nema definovany content type, predpoklada se tedy,
ze jde o US-ASCII text.

--MAIL-BOUNDARY
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Uk=E1zka obsahu. Znaky s diakritikou jsou reprezentov=E1ny
bajty. Text jde ale s lehk=FDmi obt=ED=BEemi p=F8e=E8=EDst.

--MAIL-BOUNDARY--
Paticka, ktera by take mela byt ignorovana.
```

Výpis 2.6: Ukázka použití `boundary` a `quoted-printable` dle RFC 1341

```
From: =?iso-8859-2?q?Maty=E1=B9_Hork=FD?= <xhorky31@vutbr.cz>
```

Výpis 2.7: Ukázka kódování jména dle RFC 1522

pouzdření (zpráva je nejprve podepsána a potom zašifrována), druhým způsobem je podepsání a zašifrování v jednom kroku (dle definice v RFC 4880), což může být vhodné pro systémy mimo MIME.

Všechna PGP data (podpis, zašifrovaný text) musí být *armored* (tj. nesmí být v binárním formátu) a musí být uložena ve formátu radix-64 s maximální délkou řádku 78 znaků [32].

2.5 S/MIME

Secure MIME je druhá používaná možnost jak zprávy podepisovat a šifrovat – stejně jako PGP. Na rozdíl od něj vychází ze schématu PKCS #7, podle kterého byl také

```
MIME-Version: 1.0
Content-Type: multipart/signed; micalg=pgp-sha256;
 protocol="application/pgp-signature";
 boundary="MAIL-BOUNDARY"

--MAIL-BOUNDARY
Content-Type: multipart/mixed;
boundary="FIELD-BOUNDARY"

--FIELD-BOUNDARY
Content-Type: text/plain; charset=utf-8
Content-Language: en-US
Content-Transfer-Encoding: quoted-printable

Pozdrav,
text e-mailu.
Podpis

--FIELD-BOUNDARY--

--MAIL-BOUNDARY--
```

Výpis 2.8: Ukázka části e-mailu dle RFC 3156

pojmenován MIME typ `application/pkcs7-mime`. První verze S/MIME byla vytvořena v roce 1995, nejnovější 4.0 byla vydána v dubnu 2019 [33].

Aby bylo možné ochránit hlavičky (adresáti nebo předmět), S/MIME povoluje celou MIME zprávu zabalit do formátu `message/rfc822` – takovou ochranu PGP neumožňuje. S/MIME objekty se mohou vrstvit, čímž lze zajistit podepsání, šifrování i kompresi [33].

Když je zpráva nejprve podepsána, autoři jsou bezpečně skryti. Když je zpráva nejprve zašifrována, podpisy jsou viditelné, ale zase je možné ověřit zprávu bez nutnosti znát privátní klíče. Dále dokumentace uvádí:

> *There are security ramifications related to choosing whether to sign first or encrypt first. A recipient of a message that is encrypted and then signed can validate that the encrypted block was unaltered but cannot determine any relationship between the signer and the unencrypted contents of the message. A recipient of a message that is signed and then*

encrypted can assume that the signed message itself has not been altered but that a careful attacker could have changed the unauthenticated portions of the encrypted message [33, str. 37].

```
Content-Type: multipart/signed; micalg=sha-256;
  boundary="FIELD-BOUNDARY";
  protocol="application/pkcs7-signature"

--FIELD-BOUNDARY

This is some sample content.
--FIELD-BOUNDARY
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

MIIBJgYJKoZIhvcNAQcCoIIBFzCCARMCAQExADALBgkqhkiG9w0BBwExgf4w
gfsCAQIwJjASMRAwDgYDVQQDEwdDYXJsUlNBAhBGNGvHgABWvBHTbi7EELOw
MAsGCWCGSAFlAwQCAaAxMC8GCSqGSIb3DQEJBDEiBCCxwpZGNZzTSsugsn+f
lEidzQK4mf/ozKqfmbxhcIkKqjALBgkqhkiG9w0BAQsEgYB0XJV7fjPa5Nuh
oth5msDfP8A5urYUMjhNpWgXG8ae3XpppqVrPi2nVO41onHnkByjkeD/wc31
A9WH8MzFQgSTsrJ65JvffTTXkOpRPxsSHn3wJFwP/atWHkh8YK/jR9bULhUl
Mv5jQEDiwVX5DRasxu6Ld8zv9u5/TsdBNiufGw==

--FIELD-BOUNDARY--
```

Výpis 2.9: Ukázka části S/MIME e-mailu dle RFC 8551 (převzato z [33, str. 29])

Kryptograficky používá S/MIME stejný princip jako PGP. Symetricky se zabezpečuje obsah, asymetricky klíč pro symetrickou šifru [34].

Rozdíl je ve způsobu získání klíče – zatímco se PGP klíče vzájemně podepisují a vytváří tím síť podpisů (tzv. *The Web of Trust*), S/MIME vychází z konceptu CMS (*Cryptographic Message Syntax*) a X.509 certifikátů. Platnost S/MIME certifikátů zajišťují certifikační autority, které mohou být také zodpovědné např. za vydávání TLS certifikátů.

PGP je systém rychle nastavitelný a k jeho zprovoznění není třeba třetí strana – uživatel si vytvoří identitu, vystaví svůj veřejný klíč a může ho začít využívat za předpokladu, že si tento veřejný klíč adresáti přidají do svých klíčenek. Pro získání S/MIME certifikátu je tradičně potřeba o něj požádat certifikační autoritu (nebo

jinou entitu ve stromu důvěry), adresáti však již nemusí provádět další kroky, pokud jejich e-mailoví klienti mají daný kořenový certifikát. Proto se S/MIME využívají spíše ve firemním prostředí.

2.6 Distribuce e-mailů protokolem SMTP

První dokument stanovující formát SMTP (Simple Mail Transfer Protocol) byl vydán v roce 1981 jako RFC 788. Jde o plně textový protokol využívající klíčová slova a stavové kódy; konec e-mailu je značen řádkem, jehož jediným obsahem je tečka. Aby svou zprávu uživatel nemohl ukončit omylem, při odesílání se před všechny řádky začínající tečkou přidá tečka druhá, příjemce zas takovou tečku (která je následována dalšími znaky) odebere [35].

Krom prostého zasílání je také definováno přeposílání (návratová hodnota `251 User not local`), ověření uživatele (resp. zjištění existence účtu, příkaz `VRFY`). Tato verze protokolu vznikla ještě před existencí doménových jmen (a v ukázkách je nevyužívá) a zavádí nové názvosloví (odesílatel, příjemce, odpověď, ...).

Aktualizace, RFC 821 z roku 1982, již v ukázkách používá doménu `.ARPA`, výraz `host` nahrazuje výrazem `domain` nebo zmiňuje využívání DNS záznamu `MX` [36].

K velké změně došlo v roce 2001 vydáním RFC 2821. Umožňuje použití SMTP rozšíření indikováním zprávou `EHLO` (Extended HELO, původní zpráva `HELO` je zachována pouze pro zpětnou kompatibilitu). Vynucuje použití doménových jmen (pomocí DNS záznamů `MX`, `A` a `CNAME`) a lokální směrování zakazuje. Specifikuje použití hlavičky `Received` (a její přidání každým serverem v prostředí internetu, který zprávu předává nebo přijímá). Řeší také možné problémy se zacyklováním, a to počítáním `Received` hlaviček a zastavením při jejich vysokém počtu (např. 100) [37].

Kapitola 7.1 tohoto dokumentu popisuje bezpečnost komunikace:

> *Constructing such a message so that the "spoofed" behavior cannot be detected by an expert is somewhat more difficult, but not sufficiently so as to be a deterrent to someone who is determined and knowledgeable. Consequently, as knowledge of Internet mail increases, so does the knowledge that SMTP mail inherently cannot be authenticated, or integrity checks provided, at the transport level. Real mail security lies only in end-to-end methods involving the message bodies, such as those which use digital signatures [...]* [37, str. 64].

Protokol SMTP je takto prohlášený za nezabezpečený a slouží pouze k distribuci. Odlišná `From` hlavička se objevuje i v běžném provozu, ne pouze útočném (např. pokud mají být odpovědi přeposílány na jinou adresu). Dokument žádné další problémy

nekomentuje, aby kvůli získání trochu větší bezpečnosti nebyly odstraňovány další využívané funkce.

```
S: 220 vutbr.cz Simple Mail Transfer Service Ready
C: EHLO example.com
S: 250 vutbr.cz greets example.com
S: 250 SIZE
S: 250 DSN
S: 250 HELP
C: MAIL FROM:<xhorky31@example.com>
S: 250 OK
C: RCPT TO:<xfiter00@vutbr.cz>
S: 250 OK
C: DATA
S: 354 Start mail input; end with .
C: From: Matyáš Horký <xhorky31@example.com>
C: To: Ivana Fitere <xfiter00@vutbr.cz>
C: Subject: Elektronicka posta
C:
C: Obsah zpravy prenaseny protokolem SMTP dle RFC 2821.
C: .
S: 250 OK
C: QUIT
S: 221 vutbr.cz Service closing transmission channel
```

Výpis 2.10: Přenos e-mailu dle RFC 2821. Zpráva zasílané odesílatelem jsou značeny písmenem `C` (*client*), příjemcem `S` (*server*).

Nejnovější platné RFC popisující SMTP, RFC 5321, obsahuje mnoho upřesnění a aktualizací, stejně jako tomu bylo u aktualizace formátu e-mailů v RFC 5322. Sekce 6.2 (*Unwanted, Unsolicited, and "Attack" Messages*) popisuje vzrůstající trend nevyžádané pošty. Zejména povoluje zprávu „zahodit" bez vědomí odesílatele, i když tuto praktiku nedoporučuje, protože jde o porušení dlouhé tradice, dle které je e-mail buď doručen, nebo vrácen [27, s. 72]. Také povoluje ukončit spojení ze strany serveru, pokud je chování komunikujícího klienta vyhodnoceno jako škodlivé (útoky DoS, vytěžování osobních informací zneužíváním příkazu `RCPT`). Upozorňuje na křehkost systému a na rizika, která by mohla vzniknout přílišným filtrováním doručované pošty [27, s. 78].

2.7 EFAIL – Útok na strukturu MIME s OpenPGP

Chyba EFAIL byla zveřejněna v květnu 2017 a dotkla se OpenPGP i S/MIME. Dostala označení CVE-2017-19688 (PGP) a CVE-2017-19689 (S/MIME). Oběma bylo přiřazeno CVSS skóre[1] 5.9 [38]. Tato kapitola se zabývá především problémem s PGP, princip v S/MIME je velmi podobný.

Aby útočník získal zašifrovaná data, zneužívá aktivní obsah HTML e-mailů (obrázky nebo styly) zasláním HTTP požadavku e-mailovým klientem na útočníkem kontrolovanou URL. Nejprve potřebuje přístup k zašifrovanému e-mailu (naslouchání na síti, kompromitace schránek, serverů nebo záloh). Zachycený e-mail změní způsobem popsaným níže a odešle ho oběti. E-mailový klient oběti obsah dešifruje a stáhne externí obsah, čímž dojde k odeslání dat útočníkovi [39].

EFAIL se skládá ze dvou způsobů jak zašifrovaný obsah získat.

První je útok na operační mód CFB. Protože zašifrovaný obsah téměř vždy začíná textem `Content-Type: multipart/mixed`, útočník zná alespoň jeden blok nešifrovaného textu. S touto znalostí lze vytvořit umělý *plaintext* blok, který obsahuje pouze nuly. Tento blok se nazývá *CFB gadget* a může být vkládán do původní zprávy, čímž vytvoří HTML tag `img`.

CFB gadget může být matematicky popsán jako $((C_i, C_{i+1}), P_i)$, kde (C_i, C_{i+1}) je pár šifrovaných bloků a P_i je blok čistého textu. S tímto *gadgetem* je možné transformovat P_i do jakéhokoliv *plaintextu* P_c nahrazením C_i s $X = C_i \oplus P_i \oplus P_c$.

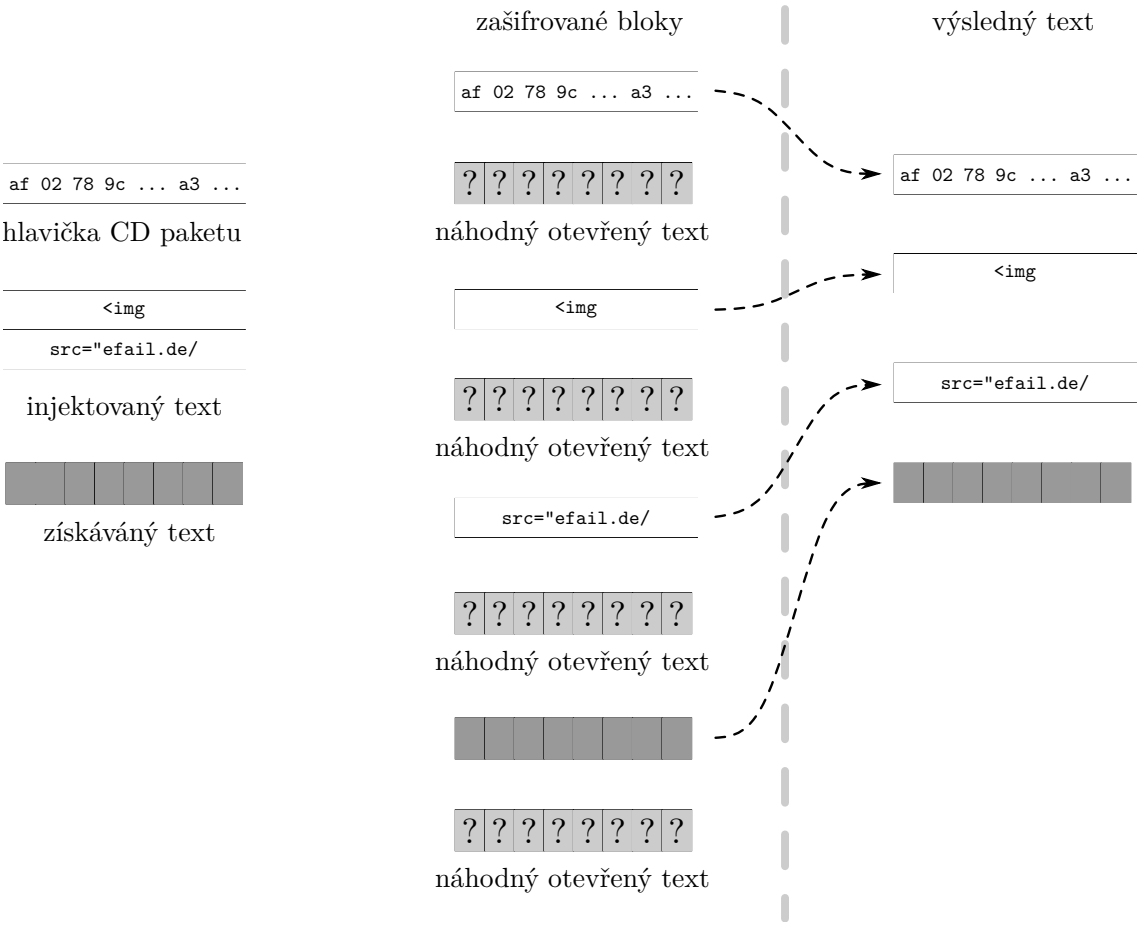
Vkládání *gadgetů* vytváří sekvenci bloků náhodných dat a uměle vloženého textu. Aby byl útok úspěšný, útočník musí zajistit, aby byla nekontrolovatelná náhodná data ignorována. Řešením je zakomentování nebo vložení do atributů HTML elementů, které budou při zobrazení ignorovány.

Tato metoda je mnohem účinnější při použití S/MIME, protože OpenPGP využívá MDC (*Modification Detection Code*), který takovou manipulaci dokáže detekovat. PGP také na text před jeho zašifrováním aplikuje kompresi, což znesnadňuje injekci paketů.

Druhý útok přímou exfiltrací útočí na zranitelnosti e-mailových klientů. Útočník vytvoří novou *multipart* zprávu (viz ukázku 2.11). První část e-mailu obsahuje otevření `img` tagu, druhá samotný obsah a třetí ukončení obrázku. E-mailový klient oběti tyto tři zprávy složí do jedné a vytvoří tím jeden HTML tag s obrázekm, jehož URL je dešifrovaný obsah zprávy.

Jedním z jednoduchých řešení je vypnutí zobrazení externích zdrojů pro HTML e-maily, což se ve většině klientů již děje automaticky. Pořádnou opravou je ale aktualizace OpenPGP a S/MIME dokumentů.

[1] CVSS (Common Vulnerability Scoring System) je způsob hodnocení závažnosti bezpečnostních zranitelností.



Obr. 2.1: Schéma CFB exfiltrace (Převzato z [39, str. 555])

`python-gnupg-mail` získaná data nijak neinterpretuje, EFAIL se ho tedy přímo netýká. Jde však o důležitý milník v historii bezpečnosti elektronické pošty, proto byl do práce zařazen.

```
From: attacker@example.com
To: victim@example.com
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/html

<img src="http://efail.de/
--BOUNDARY
Content-Type: application/pkcs7-mime; smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxhhHXMIIB0wIB...
--BOUNDARY
Content-Type: text/html
">
--BOUNDARY--
```

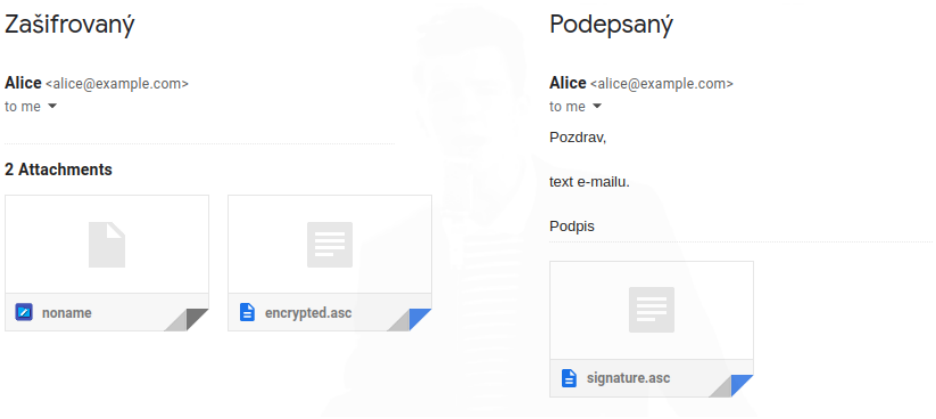
Výpis 2.11: Přímá exfiltrace. Ukázka je převzata z [39, str. 557] a zachycuje S/MIME, útok na OpenPGP je však zcela stejný.

3 PGP v praxi

Jak bylo ve dvou předcházejících kapitolách vysvětleno, použití OpenPGP v elektronické poště je zcela dobrovolné. Přímou podporu nabízí pouze část programů, pro některé existují doplňky od nezávislých autorů.

3.1 Webové klienty

Webové prohlížeče z bezpečnostních důvodů navšívené stránky maximálně izolují od systému, ve kterém prohlížeč běží. Pošta v prohlížeči tedy nemá žádný způsob přístupu k nainstalovanému PGP programu ani k privátním klíčům, které jsou nutné k úspěšnému dešifrování. Pošta (např. Gmail na obrázku 3.1) je zde zobrazena dle pravidel určených RFC 4880: jsou zobrazeny části, které program dokáže interpretovat (text v podepsaném e-mailu), neznámé MIME objekty (zašifrovaný text, podpis) jsou přiloženy pod zprávou jako přílohy.



Obr. 3.1: Zašifrovaná zpráva se skládá ze dvou MIME částí, které Gmail zobrazit nedokáže. Podepsaná zpráva se zobrazí, ověřena ale není.

3.2 Webové klienty podporující šifrování

Během posledních dvaceti let došlo k masivnímu přesunu z lokálních e-mailových klientů do prostředí webu. Používat PGP standardním způsobem v prohlížeči není možné, protože stránky nemají přístup k lokálním souborům a senzitivním informacím, jako jsou privátní klíče. Šifrován je většinou pouze přístup ke schránce, přesto ale existují systémy, které *end-to-end* zabezpečení e-mailů v prohlížeči umožňují.

3.2.1 ProtonMail

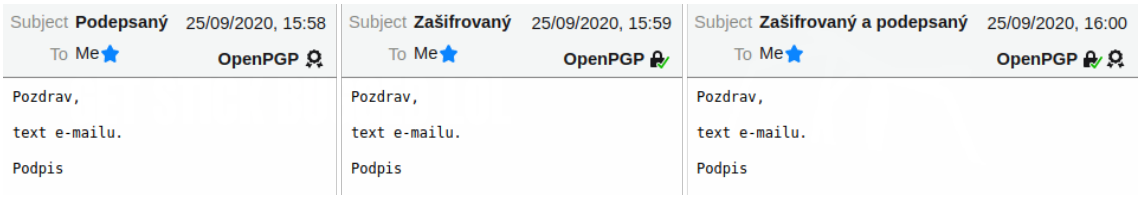
Služba ProtonMail funguje na principu OpenPGP: při vytváření nového účtu se v prohlížeči vytvoří pár RSA klíčů, které služba používá pro kryptografii. Na serveru je uložen veřejný klíč a privátní klíč zašifrovaný symetricky využitím AES [40]. *Frontend* (webmail, aplikace) je vyvíjen jako opensource, *backend* je z bezpečnostních důvodů neveřejný [41].

3.2.2 Tutanota

Druhým známým webmailem je německá Tutanota. Využívá svůj vlastní způsob šifrování obsahu, který je na PGP pouze založený (RSA, AES), díky tomu lze šifrovat kromě obsahu a příloh i jména komunikujících stran či předmět zprávy, a také lze zajistit *Perfect Forward Secrecy* [42]. *Frontend* (webmail, aplikace) je vyvíjen jako opensource [43].

3.3 Thunderbird

Mozilla Thunderbird je multiplatformní e-mailový klient vytvořený společností Mozilla Foundation a od počátku roku 2020 vyvíjený její dceřinnou společností MZLA Technologies Corporation [44]. Rozšíření Enigmail existovalo od roku 2001 a přidávalo podporu čtení a odesílání OpenPGP zpráv. Od verze 78 Thunderbird obsahuje podporu OpenPGP přímo, bez využití systémového PGP a Enigmailu [45].

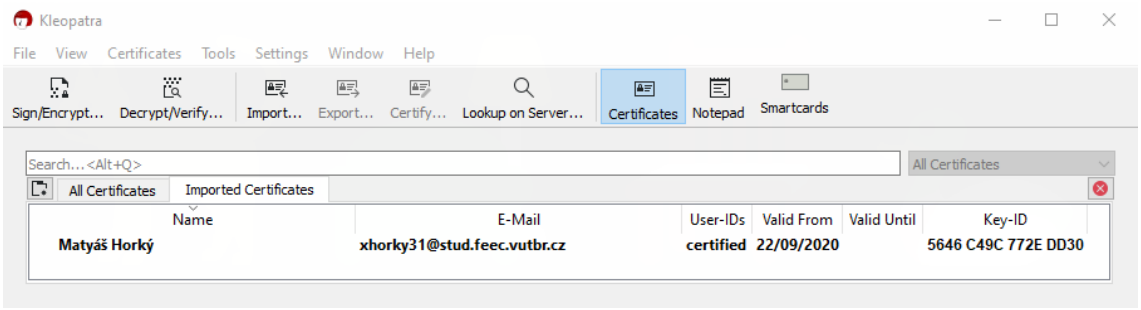


Obr. 3.2: Informace o zabezpečení je v Thunderbirdu zobrazena vpravo nad obsahem zprávy.

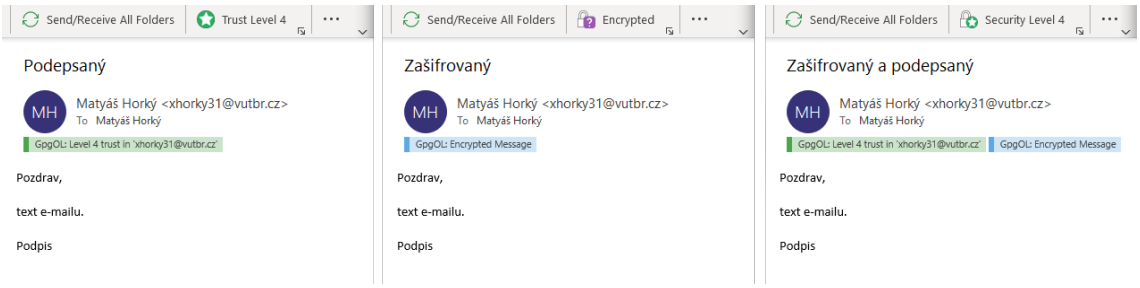
3.4 Outlook

Microsoft Outlook je e-mailový klient společnosti Microsoft nabízený v rámci sady kancelářských programů Microsoft Office 365. Výchozí instalace podporuje pouze S/MIME a Information Message Standard (v rámci Office 365 Enterprise E3) [46].

Podporu OpenPGP dodává software Gpg2win integrující nástroje Kleopatra (pro management vlastních i cizích PGP klíčů) a GpgOL (Outlook plugin) [47].



Obr. 3.3: Nástroj Kleopatra pro management PGP klíčů v systému Windows.



Obr. 3.4: Informace o zabezpečení je v Outlooku zobrazena v horním panelu.

4 Řešení

Cílem praktické části této práce bylo vytvoření knihovny pro Python umožňující ověřování digitálních podpisů PGP v e-mailové komunikaci.

4.1 Python

Python je interpretovaný vysokoúrovňový programovací jazyk. Mezi jeho výrazné znaky patří používání *whitespace* znaků pro strukturování textu v kontrastu s velkým množstvím dalších jazyků (C, Rust, Java, JavaScript), které pro strukturování kódu používají složené (případně jiné) závorky. Python podporuje objektově i funkcionálně strukturované orientované programování [48, 49].

Tab. 4.1: Podporované verze jazyka Python ke květnu 2021

Verze	Datum vydání	Konec podpory
3.6	23. 12. 2016	23. 12. 2021
3.7	27. 6. 2018	27. 6. 2023
3.8	17. 10. 2019	říjen 2024
3.9	5. 10. 2020	neoznámeno

4.1.1 Knihovna `python-gnupg`

`python-gnupg` je knihovna pro Python zajišťující komunikaci s programem `gnupg` nainstalovaným v systému. Podporuje všechny „živé“ verze Pythonu a zastaralou verzi 2.7 pro situace, kdy nasazení třetí verze Pythonu možné není [5].

S `gnupg` komunikuje prostřednictvím *shellu* a informace získává parsováním výstupu. Tento krok byl v historických verzích nedostatečně zabezpečen: v roce 2013 proto Isis Lovecruft vytvořila svou verzi s názvem `pretty-bad-protocol` [50]. Bezpečnostní problém byl v následujících letech vyřešen a používat alternativy již není třeba, také protože se jim nedostává takové pozornosti z hlediska autorů i uživatelů.

V tabulce 4.2 jsou porovnány Python knihovny pro GPG, které byly publikovány na PyPI (viz kapitolu 4.3.1). Web `pystats.org` k balíčkům ukazuje počet stažení za poslední den, týden a měsíc a `python-gnupg` je v počtu stažení řádově nejpopulárnější.

Druhým balíčkem je `gnupg`, což je ve skutečnosti stará verze `pretty-bad-protocol`, který se drží na místě čtvrtém (výrazná část těchto stažení tedy může být způsobena

chybou uživatele a nezamýšlenou instalací špatného balíčku). Na třetím místě je nezávislý balíček `PGPy`, který šifrování a ověřování provádí sám za pomoci systémových kryptografických knihoven[1].

Tab. 4.2: Porovnání Python knihoven pro práci s GPG

knihovna	**verze**	**datum vydání**	**květen 2021**
`python-gnupg`	0.4.7	11. 3. 2021	1 817 392
`gnupg`	2.3.1	7. 9. 2017	148 529
`pgpy`	0.5.4	17. 4. 2021	102 381
`pretty-bad-protocol`	3.1.1	1. 8. 2018	50 458

4.1.2 Kvalita kódu a jeho testování

Čitelný kód splňující standardy je jedním z předpokladů udržitelného programu. Velká část této odpovědnosti leží na programátorovi (rozdělení do funkcí, pojmenování proměnných, struktura kódu). Jiné vlastnosti, jakou je například formátování, lze přenechat na nástrojích statické analýzy, které mohou dodržování zažitých požadavků na kód kontrolovat.

V Python projektech se tyto vývojové nástroje standardně uvádí v textovém souboru `requirements-dev.txt`, odkud je lze nainstalovat pomocí balíčkovacího nástroje `pip`. Jejich spouštění lze zajistit například konfigurací textového editoru (při každém uložení či ručním zavolání) nebo s tzv. pre-commit hookem – ten jde jednoduše nastavit nástrojem `pre-commit`.

Dalším z nástrojů je `black`. Při spuštění načte kód souboru a zformátuje ho podle formátovacích požadavků PEP8. Díky tomuto nástroji má celý kód jednotnou formu; při práci nemusí autor přemýšlet nad formátováním jako takovým – `black` tu práci provádí za něj. Po změnách vždy dochází ke kontrole, jestli se shoduje CST (Concrete Syntax Tree) před a po spuštění, takže nemůže dojít ke změně kódu samotného, a tedy k nechtěné změně funkčnosti [51].

Sada nástrojů `flake8` provádí statickou analýzu Python kódu. Do napsaného kódu nezasahuje přímo – poskytuje informace o potenciálních problémech od chybějící definice proměnné po špatné formátování [52].

Balíček `pytest` umožňuje jednoduché a rychlé vytváření testů, které mohou kontrolovat funkčnost celku i jednotlivých součástí programu. Při změnách kódu lze tedy automaticky zjistit, že došlo ke změně chování programu [53].

[1]Data byla získána 28. května 2021 pomocí webového přístupu na `https://pypistats.org/packages/<package>`.

Python je dynamický jazyk umožňující *duck typing*[2]: atributy či funkce objektu jsou využívány bez ohledu na jeho typ. Dodání nesprávného datového typu způsobí chybu při běhu programu. Nástroj statické analýzy `mypy` z kódu čte nepovinné typové anotace a sleduje, zda jsou funkcím dodávány správné datové typy [54]. Po začlenění tohoto nástroje do systému kontrol při vývoji knihovny `python-gnupg-mail` bylo nalezeno několik chyb, které by v produkčním prostředí způsobovaly pády programu.

4.2 Knihovna pro kontrolu OpenPGP v MIME

Program `python-gnupg-mail` umožňuje načítat soubory e-mailů a procházet jejich strukturu. Třídy a funkce jsou dokumentovány a je z nich automaticky vytvářena dokumentace s pomocí programu Sphinx. Obsahuje čtyři desítky testů, které kontrolují dílčí i celkovou funkčnost kódu a zabraňují tak nechtěnému vytváření chyb ve vnitřní logice.

Diagram tříd je v příloze B na straně 59.

4.2.1 Objekty zprávy

Soubor `field.py` obsahuje dvě třídy, které reprezentují strukturu MIME e-mailu (`Field`, `MultipartField`), a dvě, které na nich staví a přidávají informace o PGP kontextu (`EncryptedField`, `SignedField`).

Třída `Field` je rodičem pro všechny ostatní objekty reprezentující informace. Přijatá data rozdělí na řádky a ty na hlavičky a tělo. Z hlaviček získá důležité informace, jako je například atribut `Content-Type`, které jsou používány pro určení typu objektů a manipulaci s nimi.

`MultipartField` je jejím potomkem. Má navíc atribut `fields`, což je seznam dalších MIME objektů, které obsahuje. `EncryptedField` a `SignedField` jsou potomky této třídy a ukládají do sebe data získaná pomocí knihovny `python-gnupg`, tedy například dešifrovaný obsah e-mailu.

Soubor `message.py` obsahuje pouze třídu `Message`. Veškerá logika zjišťování informací je spuštěna při inicializaci této třídy. Rekurzivně skenuje svůj obsah a vytváří z něj strukturu, kterou analyzuje a vyhodnocuje.

4.2.2 Parsery pro manipulaci se zprávou

Pro rozdělení zprávy bylo využito několik oddělených parserů.

[2]Název vznikl podle tzv. *testu kachnou*: „Pokud do vypadá jako kachna a kváká jako kachna, nejspíš to kachna bude.“

Prvním je třída `TextParser`. Jeho metoda `split()` se stará o rozdělování řádků na hlavičku a tělo (jsou vždy odděleny prázdným řádkem). Funkce `unfold()` odstraňuje složené řádky – prochází jeden po druhém a pokud narazí na takový, který začíná *whitespace* znakem, připojí ho k předcházejícímu. Čištění získaných řádků zajišťuje `strip_lines()`.

`HeadParser` pracuje s hlavičkami e-mailu a jednotlivých MIME částí. Metoda `get_headers()` prochází seznam řádků a vytváří z nich slovník rozdělením na prvním znaku dvojtečky, metody `get_content_type()` a `get_boundary()` získávají konkrétní typ obsahu a hraniční řetězec – oboje je nutné k úspěšnému načtení vnitřního obsahu.

Třída `MultipartParser` obsahuje jedinou metodu: `separate()`. Jejím úkolem je rozdělit tělo s primárním typem `multipart` na další MIME objekty. Využívá k tomu hraniční řetězec, který má každý `multipart` objekt definovaný.

4.2.3 Popis funkčnosti programu

Při spuštění programu z příkazové řádky (viz následující kapitolu o balíčkování) je volána funkce `main()` v souboru `main.py`. Uživatel použitím argumentů určí nastavení aplikace (jestli má být vyžadováno šifrování, jaká úroveň logování se má použít) a soubor e-mailu, který chce ověřit. Obsah EML (*Electronic Mail Format*) souboru a nastavení je předáno konstruktoru třídy `Message`, kde se provede veškeré čtení a kontrola zprávy.

Pro použití programu jako knihovny je po instalaci třeba provést import třídy `Message`, načtení kontrolovaného EML souboru a volitelně také vytvoření objektu nastavení. Když je nastavení vynecháno, použijí se výchozí hodnoty.

```
1 from gnupg_mail import Message
2
3 with open("path/to/the_email.eml", "r") as handle:
4     raw = handle.read()
5 message = Message(raw)
```

Výpis 4.1: Použití programu ve formě knihovny

Následující popis zpracování dat je bez znalosti programu relativně komplikovaný. Pro referenci a orientaci je v příloze C na straně 60 umístěn diagram.

Jako první se provádí převod textu na jednotlivé řádky se kterými program pracuje. To se provádí rozdělením textu tam, kde se vyskytuje znak nového řádku `CR` nebo `CRLF`. (Všechna relevantní RFC uvádí, že se pracuje výhradě s textem s konci řádků značenými bajty `CRLF`. Na UNIXových systémech se ale využívá pouze

`CR` a takové soubory by program nerozpoznal. Jde tedy o určitou formu abstrakce nad rozdílností operačních systémů.)

Třída `Message` je potomkem třídy `Field`. Aby dávalo smysl vysvětlení logiky, která je spuštěna po načtení zprávy, nejprve je nutné popsat chování tříd MIME objektů.

`Field` je univerzální třída do které se načítají všechny nalezené MIME objekty – některé trvale, některé pouze za účelem získání metadat objektu (`MultipartField` a další).

Důležitým atributem je privátní proměnná `_raw_lines`, která drží původní, neupravované řádky. Ty jsou totiž funkcí `TextParser.unfold()` rozloženy, aby nové řádky nezačínaly *whitespace* znakem. Na prvním prázdném řádku je pak list řádků rozdělen na hlavičku a tělo. Z řádků hlaviček je vytvořen slovník (*dictionary*) `headers`, ve kterém jsou uložena metadata MIME objektu jako typ nebo kódování. Pokud slovník neobsahuje klíč `Content-Type`, je typ objektu nastaven na `text/plain` dle RFC 2045 [55, str. 14]. Zbytek souboru pod prvním prázdným řádkem je uložen jako tělo objektu a v této třídě se s ním již nepracuje.

Třída `MultipartField` je potomkem `Field`. Její vytvoření je podmíněno hodnotou `Content-Type`: primární typ musí být `multipart`. Po své inicializaci je použita funkce `MultipartParser.separate()`, kterou se tělo zprávy rozdělí do dalších MIME objektů na základě hodnoty `boundary`, tj. hraničního řetězce MIME skupiny. Nalezené skupiny řádků jsou převedeny na MIME objekty a proces se rekurzivně opakuje, dokud jsou nacházeny další vnořené `multipart` objekty.

Syntaxe hlaviček e-mailu a MIME objektů je totožná, děděním z třídy `Field` se zabrání duplikování kódu. Zavoláním funkce `super().__init__()` třídy `Message` dojde k provolání konstruktoru mateřské třídy `Field` a k načtení metadat zprávy (hlavičky adresátů, předmět, ...) a těla, které se rekurzivně zpracuje do stromové struktury.

Pokud je zpráva zašifrovaná a/nebo podepsaná (podtypy `encrypted` a `signed`), MIME objekty jsou z atributu `fields` přesunuty do jiného s názvem `old_fields`. Díky tomu nejsou ztraceny poté, co je s OpenPGP obsahem zprávy manipulováno (např. dešifrování).

Pokud je tělo e-mailu zašifrované, MIME objekty zprávy jsou zpracovány funkcí `crypto.decrypt()` a výsledek je uložen do atributu `fields`, který byl v předešlém kroku „vyčistěn“. Obdobně jsou ověřena podepsaná data funkcí `crypto.verify()`.

Výsledek dešifrování a ověření je předán privátní funkci třídy `Message` nazvané `_resolve_gnupg()`. Zde pomocí série podmínek dochází k detekci nedostatečného výsledku, kterým může být například absence šifrování, i když je nastavením vyžadováno, nedešifrovatelný obsah nebo neplatný PGP podpis (viz diagram C.3 na straně 61). Tato funkce vrací dvojici hodnot: binární výsledek operace (`True` pokud je

zpráva v pořádku, `False` pokud došlo k problémům) a textový doprovod, který vysvětluje binární hodnotu. Hodnoty jsou uloženy do atributů zprávy `valid` a `result`.

Pokud je program spuštěn z příkazové řádky, výsledek je vypsán do konzole se stavovým kódem `0` pokud vše proběhlo v pořádku a `1` pokud nastala chyba.

Pokud je však program použit jako knihovna, data zůstávají uložena ve zprávě a jak je s nimi naloženo je v rukou programátora.

4.3 Balíčkování Python aplikací

CLI (*Command Line Interface*) je textové uživatelské rozhraní, které uživatel používá zadáváním příkazů. Od grafického se liší tím, že se k jeho ovládání využívá pouze klávesnice; na úkor strmější učící křivky umožňuje rychlejší ovládání. Chování programu je ovlivněno parametry zadanými za název spouštěného programu, což umožňuje programy používat dávkově bez přímého zásahu uživatele.

V jazyce Python se pro načítání parametrů využívá knihovna `argparse`. Po vytvoření objektu `ArgumentParser` lze přidávat argumenty voláním `add_argument()`, získané hodnoty jsou pak dostupné jako jeho atributy. `python-gnupg-mail` tento způsob načítání vstupu od uživatele získává v souboru `main.py`. Dostupné parametry a jejich popisy jsou dostupné po zavolání programu s argumentem `--help`.

```
$ gnupg-mail --help
usage: gnupg-mail [args] [file]

Decrypt e-mails and check their signatures.

Arguments:
  file  path to .eml file (pass "-" to read from stdin)
        (default: None)
  --log-level {none,error,warning,info,debug}
        select output level (default: error)
  --require-encrypted
        return non-zero code if the message can't be
        successfully decrypted (default: False)
  --require-signed
        return non-zero code if the signature can't be
        successfully verified (default: False)
```

Výpis 4.2: Nápověda programu

Aby byl skript spustitelný jako jakýkoliv jiný program, je třeba ho nainstalovat. Pro potřeby distribuce a instalce je třeba vytvořit a vyplnit soubor `setup.cfg` informacemi o balíčku (název, licence), jeho autorovi (jméno a e-mail) a další. Pro potřeby instalace je však důležitá část `[options]`.

Proměnná `packages` instalačnímu programu ukazuje na složky, ve kterých se nachází kód. Jediná hodnota je `gnupg_mail`, protože obsahuje veškerý kód knihovny (struktura celého kódu je dostupná na straně 58 v příloze A). Knihovna bude také zavoláním `import gnupg_mail` použitelná v dalších skriptech [56].

Dostupnost skriptu jako existujícího programu je zajištěna konfiguračním parametrem `console_scripts`, který obsahuje mapování názvu programu na skript a funkci. Zjednodušená ukázka nastavení je ve výpisu 4.3.

```
1  [metadata]
2  name = python-gnupg-mail
3  description = Verification tool for PGP-enabled e-mails
4  author = Matyáš Horký
5  license = MIT
6
7  [options]
8  packages = gnupg_mail
9  install_requires = [python-gnupg]
10
11 [options.entry_points]
12 console_scripts =
13         gnupg-mail = gnupg_mail.main:main
```

Výpis 4.3: Zjednodušená verze souboru `setup.cfg`

4.3.1 Publikace Python balíčků

Kód programu je zveřejněn jako open source pod MIT licencí. Jedná se o permisivní licenci s velmi volnými požadavky na její uživatele. Je možné ji využívat s dalšími copyleftovými licencemi, jako je například GPL, nebo v placeném software; je pouze nutné uvádět, že části programu jsou licencované jako MIT [57].

Kód je spravován verzovacím systémem `git` a je umístěn na gitlab.com[3]. Na aktualizace kódu mohou reagovat další systémy například automatickou kontrolou kódu.

Každý nový příspěvek ve větvi `main` je kontrolován systémem GitLab CI/CD, který čte nastavení ze souboru `.gitlab-ci.yml`. Automatizovaný systém zaručuje,

[3] `https://gitlab.com/matyashorky/python-gnupg-mail`

že nově začleněný kód funguje správně a nenarušuje chování. Používá stejné nástroje jaké se využívají při vývoji, zajišťuje kontrolované a reprodukovatelné prostředí: kontroluje strukturu kódu, detekuje syntaktické chyby a spouští aplikační testy.

Na změny kódu také reaguje stránka ReadTheDocs.org[4]. Repozitář si stáhne a pomocí nástroje Sphinx vytvoří dokumentaci, kterou poté nabízí na svých stránkách. Uživatelé, kteří uvažují o použití daného projektu, se tak mohou seznámit s jeho vlastnostmi a rozhraními, aniž by museli zkoumat zdrojové kódy.

[4] `https://python-gnupg-mail.readthedocs.io`

5 Závěr

Cílem bakalářské práce bylo nastudovat strukturu e-mailu, funkci PGP, jejich vzájemné propojení a na tomto základě vytvořit program schopný e-maily ověřovat.

V první kapitole je popsán ekosystém PGP. Začíná na počátku devadesátých let první verzí a přes legální spory a změny vlastníků se dostává k současné verzi PGP 5. Popisuje OpenPGP, jeho princip a algoritmy, organizaci a reprezentaci dat a jejich ukládání. Ke konci zmiňuje některé problémy a nedostatky PGP, které je při využívání systému nutné brát v potaz.

V další kapitole je rozebrán princip elektronické pošty. Nejprve její začátky v době ARPANETu a první pokusy o sjednocení zasílaných zpráv z hlediska syntaxe a struktury. Potom konec sedmdesátých let, kdy došlo ke zpřísnění a zkonkretizování hlaviček, aby bylo možné e-maily zpracovávat strojově a efektivněji. Jsou zmíněny aktuální požadavky na elektronickou poštu a také způsoby distribuce textu s diakritikou a binárních dat pomocí MIME. Ke konci je popsán způsob propojení OpenPGP a MIME pro samotné šifrování a podepisování e-mailů. Poslední část je věnována útoku EFAIL.

Třetí kapitola zmiňuje aplikace, které s elektronickou poštou a jejím kryptografickým ověřováním pracují.

Čtvrtá kapitola popisuje praktické řešení práce. Začíná popisem jazyka Python, popisuje strukturu programu `python-gnupg-mail` a systém pro jeho vývoj a distribuci na internetu.

Program dodržuje zásady jazyka Python 3, využívá objektově orientovanou strukturu pro práci s daty, obsahuje automatizované testy i publikaci nových verzí. Lze ho využít samostatně i jako knihovnu. Vstupem je soubor elektronické pošty, výstupem je sada objektů popisujících části e-mailů s přidanými atributy, především informacemi o ověření e-mailu. Je veřejně dostupný na PyPI, oficiálním softwarovém repozitáři pro Python; automaticky generovaná dokumentace je dostupná na serveru ReadTheDocs.

Literatura

[1] CERTTOOLS. *IntelMQ* [software]. [cit. 3. prosince 2020]. Dostupné z: <https://github.com/certtools/intelmq/>

[2] INTELMQ. *2.3.0 Feature release* [online]. Březen 2021 [cit. 10. dubna 2021]. Dostupné z: <https://github.com/certtools/intelmq/releases/tag/2.3.0>

[3] Python Developer's Guide. *devguide.python.org* [online]. 2020 [cit. 5. prosince 2020]. Dostupné z: <https://devguide.python.org/#status-of-python-branches>

[4] Sunsetting Python 2. *python.org* [online]. [cit. 5. prosince 2020]. Dostupné z: <https://www.python.org/doc/sunset-python-2/>

[5] SAJIP, Vinay. *python-gnupg – A Python wrapper for GnuPG.* Leden 2021 [cit. 2. února 2021]. Dostupné z: <https://docs.red-dove.com/python-gnupg/>

[6] ZIMMERMANN, Philip R. *PGP 1.0* [software]. Červen 1991 [cit. 6. prosince 2020]. Dostupné z: <https://www.pgpkeys.org/software.html>. Požadavky na systém: DOS, velikost 75 kB.

[7] DEFCONConference. A Conversation with Phil Zimmermann. Na: *YouTube* [online]. Únor 2014 [cit. 13. prosince 2020]. Dostupné z: <https://www.youtube.com/watch?v=4ww8AAkWFhM>

[8] WONG, David. *A history of end-to-end encryption and the death of PGP.* Leden 2020 [cit. 6. prosince 2020]. Dostupné z: <https://www.cryptologie.net/article/487>

[9] ZIMMERMANN, Philip R. *PGP 2.0* [software]. Září 1992 [cit. 6. prosince 2020]. Dostupné z: <https://www.pgpkeys.org/software.html>. Požadavky na systém: DOS, velikost 150 kB.

[10] DARKNET DIARIES. *Crypto Wars* [online]. RHYSIDER, Jack. Únor 2018 [cit. 13. prosince 2020]. Dostupné z: <https://darknetdiaries.com/episode/12>

[11] ZIMMERMANN, Philip R. *PGP: Source Code and Internals.* Cambridge (Massachuttes): MIT Press, 1995. ISBN 0-262-24039-4.

[12] INTERNET ENGINEERING TASK FORCE. RFC 4880: *OpenPGP Message Format* [online]. CALLAS, Jon, DONNERHACKE, Lutz, FINNEY, Hal, SHAW, David a THAYER, Rodney. Listopad 2007 [cit. 15. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc4880>

[13] PAINE, Tom. Important Information About PGP & Encryption. *Pro-Liberty* [online]. Duben 1998 [cit. 6. prosince 2020]. Dostupné z: <https://www.proliberty.com/references/pgp/>

[14] *OpenPGP* [online]. 2020 [cit. 18. listopadu 2020]. Dostupné z: <https://www.openpgp.org>

[15] OGMIOS. Web of Trust. *Wikimedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 4. května 2021]. Dostupné z: <https://commons.wikimedia.org/wiki/File:Web_of_Trust.svg>

[16] MARLINSPIKE, Moxie. *GPG And Me* [online]. Únor 2015 [cit. 7. prosince 2020]. Dostupné z: <https://moxie.org/2015/02/24/gpg-and-me.html>

[17] Statistics. *SKS Keyservers* [online]. [cit. 7. prosince 2020]. Dostupné z: <https://sks-keyservers.net/status/key_development.php>

[18] MARLINSPIKE, Moxie. *Advanced cryptographic ratcheting.* signal.org/blog [online]. Listopad 2013 [cit. 7. prosince 2020]. Dostupné z: <https://signal.org/blog/advanced-ratcheting>

[19] GREEN, Matthew. *What's the matter with PGP?* cryptographyengineering.com [online]. Srpen 2014 [cit. 7. prosince 2020]. Dostupné z: <https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp>

[20] *GnuPG* [online]. Leden 2021 [cit. 25. února 2021]. Dostupné z: <https://gnupg.org/>

[21] INTERNET ENGINEERING TASK FORCE. RFC 561: *Standardizing Network Mail Headers* [online]. BHUSHAN, Abhay, POGRAN, Ken, TOMLINSON, Ray a WHITE, Jim. Září 1973 [cit. 18. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc561>

[22] INTERNET ENGINEERING TASK FORCE. RFC 680: *Message Transmission Protocol* [online]. MYER, Theodore a HENDERSON, Austin. Duben 1975 [cit. 18. října 2020].

Dostupné z: <https://tools.ietf.org/html/rfc680>

[23] INTERNET ENGINEERING TASK FORCE. RFC 724: *Proposed Official Standard for the Format of ARPA Network Messages* [online]. POGRAN, Ken, VITTAL, John, CROCKER, David a HENDERSON, Austin. Březen 1977 [cit. 20. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc724>

[24] INTERNET ENGINEERING TASK FORCE. RFC 733: *Standard for the Format of Text Messages* [online]. CROCKER, David, VITTAL, John, POGRAN, Ken a HENDERSON, Austin. Listopad 1977 [cit. 20. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc733>

[25] INTERNET ENGINEERING TASK FORCE. RFC 822: *Standard for ARPA Internet Text Messages* [online]. CROCKER, David. Srpen 1982 [cit. 20. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc822>

[26] INTERNET ENGINEERING TASK FORCE. RFC 5322: *Internet Message Format* [online]. RESNICK, Peter. Říjen 2008 [cit. 15. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc5322>

[27] INTERNET ENGINEERING TASK FORCE. RFC 5321: *Simple Mail Transfer Protocol* [online]. KLENSIN, John. Říjen 2008 [cit. 24. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc5321>

[28] INTERNET ENGINEERING TASK FORCE. RFC 1341: *MIME: Multipurpose Internet Mail Extensions* [online]. BORENSTEIN, Nathaniel a FREED, Ned. Červen 1992 [cit. 27. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc1341>

[29] INTERNET ENGINEERING TASK FORCE. RFC 934: *Proposed Standard for Message Encapsulation* [online]. ROSE, Marshall a STEFFERUD, Einar. Září 1985 [cit. 29. října 2020]. Dostupné z: <https://tools.ietf.org/html/rfc934>

[30] INTERNET ENGINEERING TASK FORCE. RFC 1847: *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted* [online]. GALVIN, Jim, MURPHY, Sandy, CROCKER, Steve a FREED, Ned. Říjen 1995 [cit. 26. listopadu 2020].

Dostupné z: <https://tools.ietf.org/html/rfc1847>

[31] INTERNET ENGINEERING TASK FORCE. RFC 2015: *MIME Security with Pretty Good Privacy (PGP)* [online]. ELKINS, Michael. Říjen 1996 [cit. 15. října 2020].
Dostupné z: <https://tools.ietf.org/html/rfc2015>

[32] INTERNET ENGINEERING TASK FORCE. RFC 3156: *MIME Security with OpenPGP* [online]. ELKINS, Michael, TORTO, Dave, LEVIEN, Ralph a ROESSLER, Thomas. Srpen 2001 [cit. 15. října 2020].
Dostupné z: <https://tools.ietf.org/html/rfc3156>

[33] INTERNET ENGINEERING TASK FORCE. RFC 8551: *S/MIME Version 4.0 Message Specification* [online].
SCHAAD, Jim, RAMSDELL, Blake a TURNER, Sean.
Duben 2019 [cit. 23. ledna 2021].
Dostupné z: <https://tools.ietf.org/html/rfc8551>

[34] BURDA, Karel. *Kryptografie okolo nás*.
Praha: CZ.NIC, 2019. ISBN 978-80-88168-52-2.

[35] INTERNET ENGINEERING TASK FORCE. RFC 788: *Simple Mail Transfer Protocol* [online].
POSTEL, Jonathan. Listopad 1981 [cit. 5. května 2021].
Dostupné z: <https://tools.ietf.org/html/rfc/788>

[36] INTERNET ENGINEERING TASK FORCE. RFC 821: *Simple Mail Transfer Protocol* [online].
POSTEL, Jonathan. Srpen 1982 [cit. 5. května 2021].
Dostupné z: <https://tools.ietf.org/html/rfc/821>

[37] INTERNET ENGINEERING TASK FORCE. RFC 2821: *Simple Mail Transfer Protocol* [online].
KLENSIN, John. Duben 2001 [cit. 5. května 2021].
Dostupné z: <https://tools.ietf.org/html/rfc/2821>

[38] National Institute of Standards and Technology. *NVD - CVE-2017-19688*.
Únor 2019 [cit. 1. března 2021].
Dostupné z: <https://nvd.nist.gov/vuln/detail/CVE-2017-17688>

[39] PODEBNIAK, Damian, et al.DRESEN, Christian, MÜLLER, Jens, ISING, Fabian, SCHINZEL, Sebastian, FRIEDBERGER, Simon, SOMOROVSKY, Juraj, SCHWENK, Jörg. *Efail: Breaking S/MIME and OpenPGP Email*

Encryption using Exfiltration Channels. Baltimore, Maryland, USA: USENIX, srpen 2018 [cit. 5. března 2021]. ISBN 978-1-939133-04-5.
Dostupné z: <https://efail.de>

[40] PROTONMAIL. *Zero Access to User Data* [online]. [cit. 6. dubna 2021].
Dostupné z: <https://protonmail.com/security-details>

[41] PROTONMAIL. *ProtonMail Web Client* [software]. [cit. 6. dubna 2021].
Dostupné z: <https://github.com/ProtonMail/WebClient>

[42] TUTANOTA. *Encrypted email, free & easy* [online]. [cit. 6. dubna 2021].
Dostupné z: <https://tutanota.com/encrypted-email/>

[43] TUTAO. *Tutanota* [software]. [cit. 6. dubna 2021].
Dostupné z: <https://github.com/tutao/tutanota>

[44] KEWISCH, Philipp. *Thunderbird's New Home* [online].
Leden 2020 [cit. 3. dubna 2021]. Dostupné z:
<https://blog.thunderbird.net/2020/01/thunderbirds-new-home>

[45] mozilla.org. *OpenPGP in Thunderbird - HOWTO and FAQ* [online].
Leden 2020 [cit. 3. dubna 2021].
Dostupné z: <https://support.mozilla.org/en-US/kb/openpgp-thunderbird-howto-and-faq>

[46] microsoft.com. *Encrypt email message* [online]. [cit. 18. dubna 2021].
Dostupné z: <https://support.microsoft.com/en-us/office/encrypt-email-messages-373339cb-bf1a-4509-b296-802a39d801dc>

[47] GnuPG contributors. *Gpg4win 3.1.15* [software].
Leden 2021 [cit. 18. dubna 2021]. Dostupné z: <https://gpg4win.org>.
Požadavky na systém: Windows 7 a novější, Microsoft Outlook 2010 a novější, velikost 29 kB.

[48] SUMMERFIELD, Mark. *Python 3: výukový kurz*.
Brno: Computer Press, 2010. ISBN 978-80-251-2737-7.

[49] python.org. *The Python Wiki.* [online]. [cit. 7. prosince 2020].
Dostupné z: <https://wiki.python.org/main/FrontPage>

[50] LOVECRUFT, Isis. *pretty-bad-protocol* Říjen 2018 [cit. 4. února 2021].
Dostupné z: <https://github.com/isislovecruft/python-gnupg>

[51] LANGA, Łukasz et al. *black 20.8b1* [software].
Srpen 2020 [cit. 7. prosince 2020].
Dostupné z: <https://github.com/psf/black>

[52] CORDASCO, Ian et al. *flake8 3.8.4* [software].
Říjen 2020 [cit. 7. prosince 2020].
Dostupné z: <https://gitlab.com/pycqa/flake8>

[53] KREKEL, Holger et al. *pytest 6.1.2* [software].
Říjen 2020 [cit. 7. prosince 2020].
Dostupné z: <https://github.com/pytest-dev/pytest>

[54] LEHTOSALO, Jukka et al. *mypy 0.812* [software].
Únor 2021 [cit. 28. března 2021].
Dostupné z: <https://github.com/python/mypy>

[55] INTERNET ENGINEERING TASK FORCE. RFC 2045:
Internet Message Bodies [online]. FREED, Ned, BORENSTEIN, Nathaniel.
Listopad 1996 [cit. 10. dubna 2021].
Dostupné z: <https://tools.ietf.org/html/rfc2045>

[56] Python Packaging Authority. *setuptools* [online].
Únor 2021 [cit. 9. února 2021].
Dostupné z: <https://setuptools.readthedocs.io>

[57] Open Source Inititative. *The MIT Licence.* [online]. [cit. 9. února 2021].
Dostupné z: <https://opensource.org/licenses/MIT>

Seznam symbolů a zkratek

CERT Community Emergency Response Team

CSIRT Computer Security Incident Response Team

CLI Command Line Inteface

CMS Cryptographic Message Syntax

CRLF Carriage Return, Line Feed

CVSS Common Vulnerability Scoring System

EFF Electronic Frontier Foundation

EML Electronic Mail Format

FTP File Transfer Protocol

IETF Internet Engineering Task Force

GNU GNU is not Unix

MIME Multipurpose Internet Mail Extensions

RFC Request for Comments

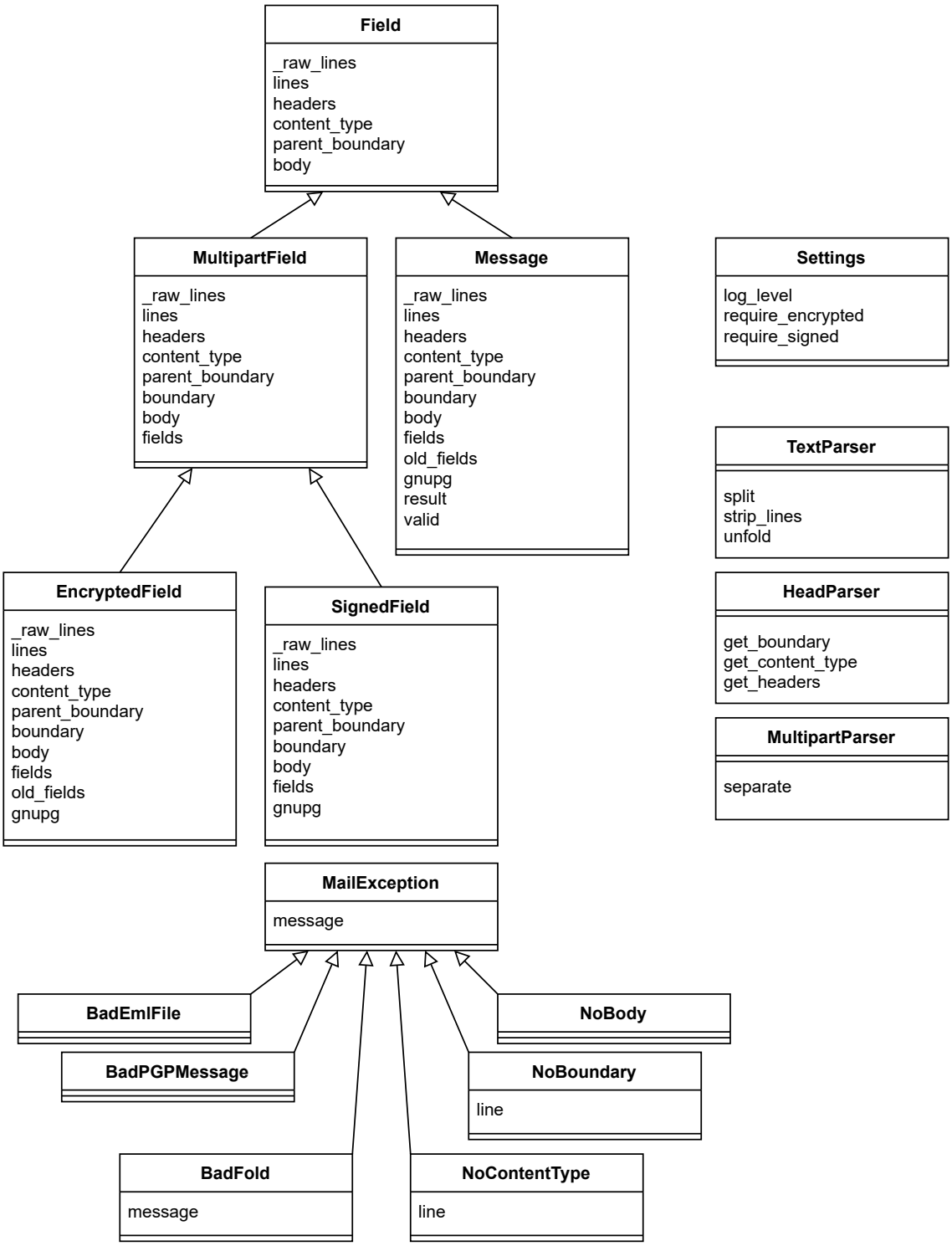
SMTP Simple Mail Transport Protocol

A Obsah elektronické přílohy

Následující výpis popisuje stav knihovny `python-gnupg-mail` ve verzi 0.3.0. Program je psán v jazyce Python 3 a je spustitelný na všech kompatibilních verzích od Pythonu 3.6 výše.

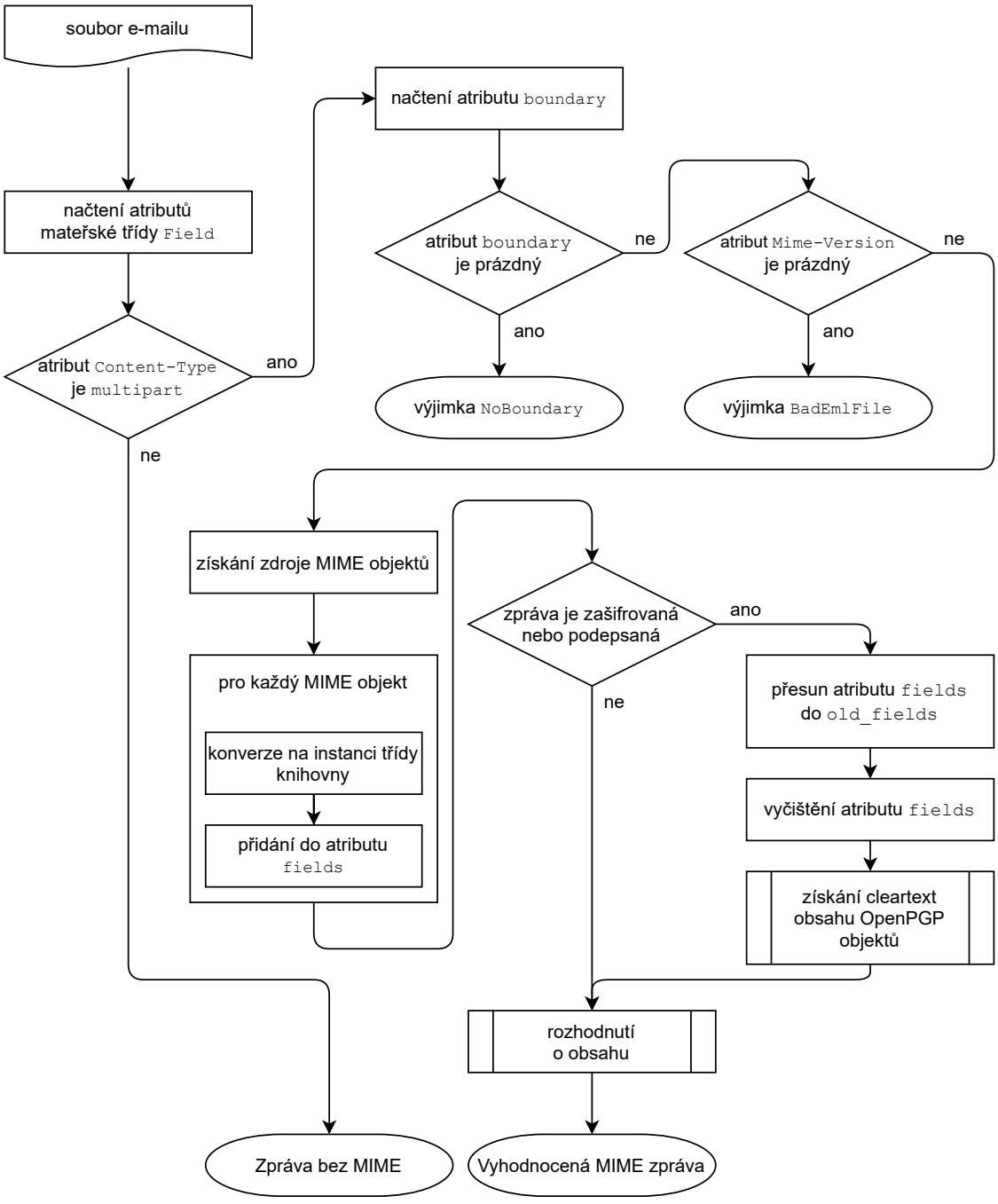
- `/` ..... kořenový adresář
  - `docs/` ..... soubory dokumentace pro ReadTheDocs
  - `files/` ..... dodatkové soubory nevyžadované knihovnou
    - `private.pgp` ..... privátní PGP klíč pro účely knihovny
    - `public.pgp` ..... veřejný PGP klíč pro účely knihovny
  - `gnupg_mail/` ..... zdrojový kód samotné knihovny
    - `__init__.py` ..... definice balíčku a jeho verze
    - `crypto.py` ..... funkce pro napojení na `python-gnupg`
    - `exception.py` ..... objekty chyb knihovny
    - `field.py` ..... objekty MIME obsahu
    - `main.py` ..... hlavní spustitelný soubor
    - `message.py` ..... objekt zprávy
    - `parser.py` ..... parsery
    - `settings.py` ..... objekt nastavení programu
  - `tests/` ..... soubory pro pytest
    - `files/` ..... ukázkové soubory e-mailů
  - `.flake8` ..... konfigurační soubor pro nástroj `flake8`
  - `.gitlab-ci.yml` ..... konfigurační soubor pro GitLab CI
  - `.mypy.ini` ..... konfigurační soubor pro nástroj `mypy`
  - `.pre-commit-config.yaml` ..... konfigurační soubor pro nástroj `pre-commit`
  - `DEVELOPMENT.md` ..... informační soubor pro vývojáře
  - `LICENSE` ..... soubor obsahující licenční podmínky použití programu
  - `README.md` ..... hlavní soubor informující o použití knihovny
  - `pyproject.toml` ..... konfigurační soubor pro nástroj `black`
  - `requirements-dev.txt` ..... seznam balíčků nutných pro vývoj
  - `requirements.txt` ..... seznam balíčků nutných pro spuštění
  - `setup.cfg` ..... konfigurační soubor pro `setuptools`
  - `setup.py` ..... konfigurační skript pro `setuptools`

B Diagram tříd programu

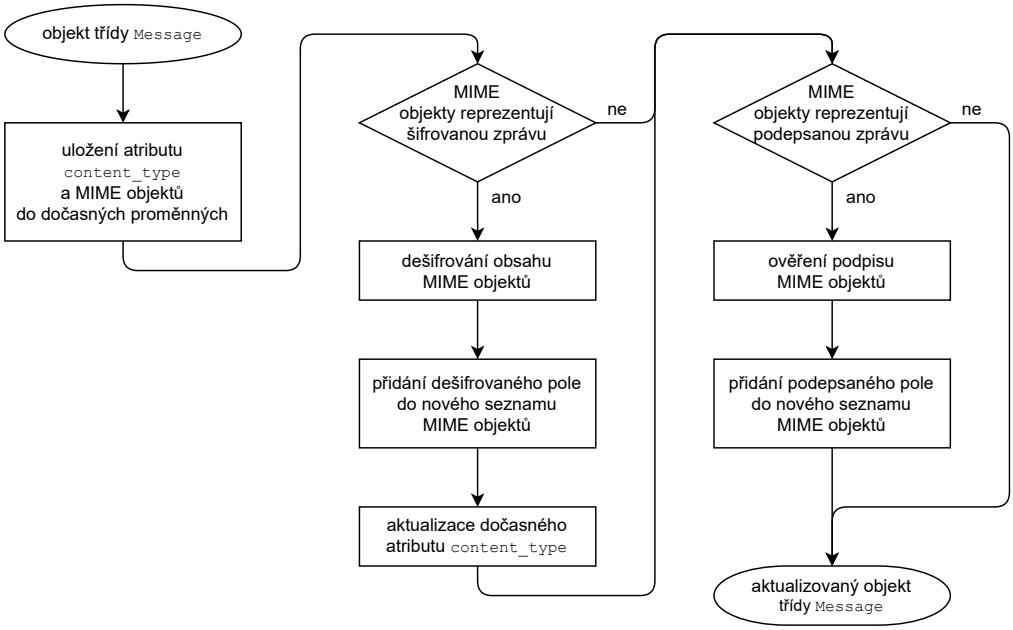


Obr. B.1: Diagram tříd

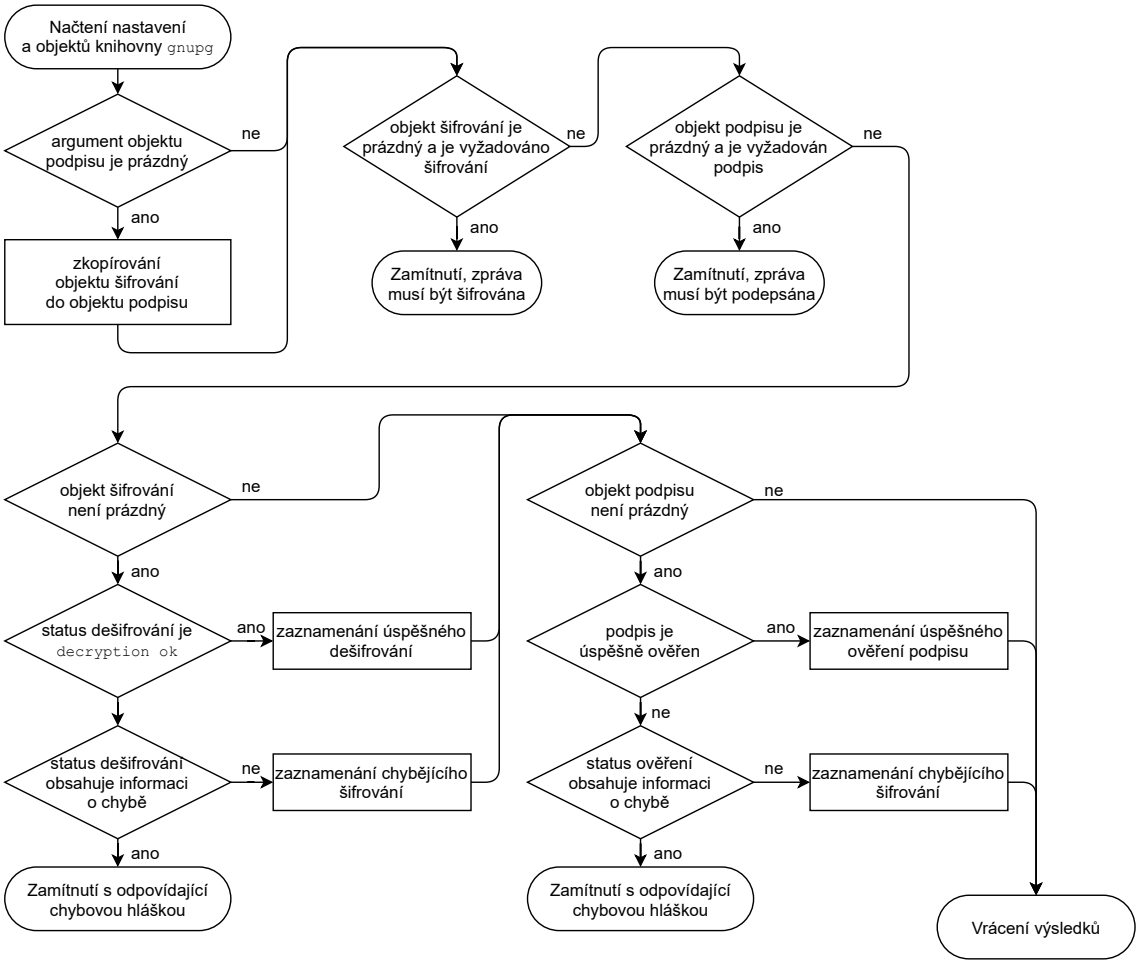
C Vývojový diagram programu



Obr. C.1: Vývojový diagram práce programu



Obr. C.2: Vývojový diagram práce s OpenPGP obsahem



Obr. C.3: Vývojový diagram získání výsledku ověření