VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií


# Kryptografie

Implementace hybridního šifrování

**Marek Žiška** **(xziska03)**

30. apríla 2023

# Obsah

# 1 Úvod a motivácia

Táto práca práca vznikla ako projekt do predmetu "Kryptografieä je zameraná na štúdium a analýzu algoritmov použitých v hybridnej kryptografii následne implementácia programu schopného demonštrovať hybridnú kryptografiu medzi klientom a serverom. Pre pochopenie kontextu bolo nutné naštudovať rôzné štúdie zaoberajúce sa hybridnou kryptografiou a algoritmami potrebné na jej implementáciu.

# 2 O realizácií projektu

Projekt bol realizovaný na OS Ubuntu 22.04 a overený na školskom serveri merlin.fit.vutbr.cz. Program využíval na implementáciu hybridnej kryptografie niekoľko knižníc:

- *os*- používa sa pre rôzne úlohy súvisiace s operačným systémom
- *time* – používa sa na funkcie súvisiace s časom, ako je spánok
- *argparse* - používa sa na analýzu argumentov príkazového riadka
- *socket* - slúži na vytvorenie sieťového socketu pre komunikáciu medzi klientom a serverom
- *Crypto.PublicKey* – používa sa na generovanie a spracovanie verejných a súkromných kľúčov RSA
- *Crypto.Cipher.AES* – používa sa na symetrické šifrovanie a dešifrovanie
- *hashlib* – používa sa na hašovacie funkcie ako md5 a sha256

Komunikácia klient-server je realizovaná hybridným prístupom. Klient vygeneruje náhodný kľúč AES pre symetrické šifrovanie a pomocou tohto kľúča zašifruje správu. Kľúč AES sa potom zašifruje pomocou verejného kľúča RSA prijímatela. Správa a zašifrovaný kľúč AES sa potom odošlú na server. Na strane servera sa šifrovaný kľúč AES dešifruje pomocou súkromného kľúča RSA servera a správa sa dešifruje pomocou kľúča AES. Server potom overí podpis pomocou verejného kľúča RSA odosielateľa. Celkovo projekt úspešne implementoval hybridnú kryptografickú komunikáciu z klienta na server, čím sa zabezpečila dôvernosť a integrita komunikácie.

# 3 Vytvaranie kľúčov a ich použitie

### 3.0.1 AES

AES kľúč sa generuje pomocou funkcie `generate_octet_string` a vbudovanej funkcii na generovanie náhodnych čísiel `os.urandom(length)`. Dĺžku kľúča som zvolil 128 bitov, pretože je to jedna zo štandardných dĺžok AES kľúčov, pričom už sama o sebe poskytuje dostatočnú bezpečnosť voči bruteforce útokom. Dĺžka kľúča 256 bitov je značne bezpečnejšia voči bruteforce útokom, ale je to na úkor výkonnosti. Na symetrické šifrovanie využívam implementáciu triedy AES z knižnice `Crypto.Cipher`, ktorá poskytuje metódu `new` na generovanie nov´ho šifrovacieho objetu. Ako režim šifry AES som zvolil Cypher Block Chaining (CBC), predaním príslušného argumentu metóde `new`. Keďže tento mód pracuje s inicializačným vektorom, ktorý si vytvorím rovnakou metódou `generate_octet_string` a dĺžku mu stanovím na 16bytov. Zarovnanie správy na násobok dĺžky AES kľúča vykonávam s použitím funkcie `generate_octet_string_except_zero` a 2 nulami, kde prvá nula označujú začiatok *paddingu* a druhá oddelluje náhodne generovanú sekvenciu od správy. Veľkosť generovanej sekvencie je stanovená veľkosťou kľúča AES a veľkosťou správy

+ dva oddelovacie znaky modulo veľkosť AES kľúča. Následne prebieha šifrovanie správy pomocou šifrovacieho objektu AES a jeho metódy `encrypt`. Pri dešifrovaní je znova potrebné použiť rovnaký IV a AES kľúč ako pri šifrovaní, preto posielam šifrovaný AES kľúč spolu s inicializačným vektorom(šifrovaný asymetricky s verejným kľúčom prijímateľa/servera). Keďže je balíček (správa + cypher MD5 hash) šifrovaný symetrickou šifrou a balíček (IV + AES key) symetrickou šifrou s verejným kľúčom prijímateľa, je zaručená dôvernost, pretože len prijímateľ vie dešifrovať AES kľúč a tým dešifrovať správu. Server po prijatí balíčka oddeluje časti šifrované asymetrickým šifrovaním(AES + IV) a symetrickým šifrovaním (správa + cypher MD5 hash), pomocou znalosti že výstupná veľkosť RSA šifry je 256bytov. Pri dešifrovaní AES kľúča a IV využíva svoj privátny kĺúč a po ich získaní dešifruje balíček (správa + cypher MD5 hash), s tým že na základe znalosti oddelenia paddingu od obsahu nulou odstraňuje padding.

#### 3.0.2 RSA

AES kľúč sa generuje pomocou funkcie `generate_or_load_RSA_keypair` ktorá buď načíta kľúče zo súbora alebo ich generuje `RSA.generate`. Dĺžku kľúča som zvolil ako 2048 bitov, ktorá je ešte aj dnes štandardizovaná, aj keď už aj použitie väčšieho kľúča 4096 začína byť populárne. Obsah šifrovaný RSA je zarovnávaný na veľkosť kĺúča a je využitý schéma *PKCS #1 v1.5*, ktorá podobne ako v mojom prípade pri AESu oddeluje správu a padding nulou. S tým rozdielom že na začiatok sa vkladá 0 a 2. Na šifrovanie sa využíva funkcia `pow`, ktorá na základe kombinácie verejného, privátneho a modula RSA objektu šifruje alebo dešifruje správu. Pri odosielaná AES kľúča sa šifruje s verejným kľúčom prijímateľa, takže len prijímateľ vie dešifrovať kĺúč symetrickej kryptografie. A pri šifrovaní MD5 charakteristiky správy využíva privátny kľúč odosielateľa. Na strane servera je proces vykonaný opačne, svojim privátnym získa AES kľúč a IV, a verejným odosielateľa overuje integritu správy porovnaním MD5 charakteristík.

## 4 Klientská strana

Trieda `MyClient` je Pythonu trieda predstavujúca klienta v aplikácii klient-server. Trieda má metódu `__init__`, ktorá inicializuje atribúty ako číslo portu, IP adresy, verejných a súkromných kľúčov RSA, verejného kľúča RSA prijímača a taktiež generuje nové AES kľúče, ktorý sa bude používať na symetrické šifrovanie správ medzi klientom a serverom.

Trieda `MyClient` má metódu `run`, ktorá pripojí klienta k serveru a čaká na vstup od používateľa. Keď dostane od používateľa nový vstup/správu, funkciou `rsa_sign_md5_with_padding` správu zahašuje a vypočíta pomocou súkromného RSA kľúča klienta jej podpis. Potom vytvorí náhodný inicializačný vektor (IV) pre danú reláciu a zašifruje správu a podpis pomocou symetrického šifrovania s kľúčom AES a IV volaním metódy `symmetric_encryption`. Po zašifrovaní správy a podpisu trieda `MyClient` zašifruje kľúč AES a IV verejným kľúčom RSA prijímateľa a odošle serveru zašifrovaný balík (zašifrovaný kľúč AES a IV a zašifrovanú správu a podpis). Ak klient vykoná počas behu programu `KeyboardInterrupt` (napríklad ak používateľ stlačí `Ctrl+C`), klient sa zastaví a končí. Metóda `create_new_aes` je pomocná metóda, ktorá generuje nový kľúč AES, ktorý sa používa pre každú novú správu odoslanú na server.

## 5 Strana serveru

Trieda `MyServer` predstavuje server, ktorý počúva prichádzajúce spojenia a komunikuje s klientmi. Konštruktor má dva voliteľné parametre: port, ktorý určuje číslo portu, na ktorom server počúva, a `ip_addr`, ktorý určuje adresu IP, na ktorú sa server viaže. Trieda `CryptoUtils` je vytvorená a

používa sa na generovanie alebo načítanie RSA kľúčov pre server a klienta, ako aj na šifrovanie a dešifrovanie správ.

Metóda `run()` sa volá po inicializácií atribútov servera a riadi beh serveru a čaká ba prichádzajúce pripojenia. Po nadviazaní spojenia servera a klienta, server príjma šifrovanú správu a rozdeluje ju na dve časti: kľúč AES zašifrovaný RSA a inicializačný vektor (IV) a samotnú zašifrovanú správu AES. Kľúč RSA sa dešifruje súkromným kľúčom servera a výsledný kľúč AES a IV sa použije na dešifrovanie správy. Dešifrovaná správa je rozdelená na dve časti: podpis a obsah. Obsah je hešovaný pomocou MD5 a podpis je overený pomocou verejného kľúča RSA klienta. Ak je podpis platný, server vytlačí správu oznamujúcu, že integrita správy nebola narušená. Ak sa počas spracovania správy vyskytnú nejaké chyby, zachytie sa o vypíšu na výstup.