Zadanie 2 – KRYPTOGRAFIA

Vypracovali: Samuel Michalčík, Marek Štrba

1. Inštalácia

Na implementáciu webového servera sme použili knižnicu Flask a SQLAlchemy pre podporu databáz.

- pip install flask flask-sqlalchemy

Po krátkom prieskume kryptografických knižníc sme sa rozhodli použiť knižnicu s názvom Cryptography, najmä kvôli podpore rôznych kryptografických operácií, algoritmov a protokolov. Pomocou tejto knižnice vieme riešiť úlohy využívajúce symetrické aj asymetrického šifrovanie, hashovanie a digitálne podpisy a zabezpečiť tak prenos dát cez nezabezpečenú sieť, či overiť pravosť a integritu dát. Knižnica je populárna aj v komunite, je jedna z najpoužívanejších s dobre vypracovanými návodmi na použitie.

- pip install cryptography

2. Implementácia úloh 2 - 4

V projekte sa v úlohách 2 až 4 zameriavame na kombináciu symetrického a asymetrického šifrovania. Cieľom je pomocou API volaní simulovať efektívne zašifrovanie a bezpečné doručenie respektíve prenos po sieti od odosielateľa príjemcovi.

Súbor je zašifrovaný pomocou AES s náhodne vygenerovaným symetrickým kľúčom, ktorý sa používa na šifrovanie aj dešifrovanie zároveň. AES sme vybrali najmä kvôli efektivite šifrovania veľkých súborov a s použitím 256-bit kľúča a 128bitového inicializačného vektora by malo ísť o jednu z najbezpečnejších (aj postkvantových) metód. Inicializačný vektor zabezpečuje náhodnosť, čo znamená, že aj ten istý plaintext zašifrovaný niekoľkokrát, bude mať rôznu podobu v zašifrovanej forme. Pri šifrovaní súboru pomocou AES využívame CFB mód, ktorý umožňuje šifrovať dáta po bajtoch.

RSA v praxi slúži na šifrovanie symetrických kľúčov (menších dát) a poskytuje systém na generovanie prepojených kľúčových párov, respektíve bezpečnú výmenu kľúčov. RSA je jedným z najpoužívanejších a najdôkladnejšie preskúmaných asymetrických algoritmov a použitý 2048-bit kľúč poskytuje dostatočnú úroveň bezpečnosti. Verejný kľúč je odvodený od privátneho kľúča a sú matematicky prepojené. Privátny kľúč je odoslaný používateľovi v kódovaní DER (binárny formát) v štandardnom formáte PKCS8 pre ukladanie privátnych kľúčov. Privátny kľúč sme nastavili bez šifrovania (NoEncryption()),

čo znamená, že kľúč je uložený v nešifrovanej podobe bez hesla – v praxi je vhodné použiť zabezpečenie privátneho kľúča. Verejný kľúč je uložený do databázy v kódovaní PEM (textový formát).

Kombináciou privátneho kľúča a verejného kľúča spolu so symetrickým kľúčom vytvárame bezpečný spôsob, ako zašifrovať dáta. AES-om vygenerovaný symetrický kľúč, ktorým je zašifrovaný daný súbor, zašifrujeme verejným RSA public kľúčom príjemcu. Pri (de)šifrovaní symetrického kľúča verejným kľúčom príjemcu používame aj OAEP padding, ktorý predstavuje mechanizmus na zvýšenie bezpečnosti (využíva masku a hashovaciu funkciu). Dešifrovanie symetrického kľúča je možné pomocou privátneho kľúča príjemcu.

PS: Curl sme vykonávali cez shell v pycharme

- /api/gen/<user> ['GET'] je request, ktorý slúži na generovanie kľúčového páru pre používateľa <user>, verejný kľúč v PEM (Base64) kódovaní uloží do databázy a privátny kľúč vráti ako response v podobe binárneho súboru
    - curl 127.0.0.1:1337/api/gen/upb -OutFile upb.key

- /api/encrypt/<user> ['POST'] je request na zašifrovanie súboru pre používateľa <user>, ktorý je prijímateľom tohto súboru. Vstupom je súbor, ktorý sa má zašifrovať, v našom prípade zadanie2 v pdf formáte. Súbor je zašifrovaný symetrickým kľúčom pomocou AES a verejným kľúčom daného usera, pre ktorého je súbor. Response je v podobe balíku, ktorý obsahuje zašifrovaný súbor encrypted.bin, zašifrovaný symetrický kľúč a inicializačný vektor.
    - curl.exe --% -X POST 127.0.0.1:1337/api/encrypt/upb -H "Content-Type: application/octet-stream" --data-binary @zadanie2.pdf --output encrypted.bin

- /api/decrypt ['POST'] je request na dešifrovanie súboru, ktorý je na vstupe aj s privátnym kľúčom prijímateľa. Zašifrovaný symetrický kľúč sa pomocou privátneho kľúča prijímateľa dešifruje a následne sa dešifruje encrypted.bin súbor. Je potrebné zachovať rovnaké kódovania, formáty a paddingy. Response je v podobe dešifrovaného decrypted.pdf súboru, ktorý je totožný s pôvodným vstupným súborom zadanie2.pdf.
    - curl.exe --% -X POST 127.0.0.1:1337/api/decrypt -F "file=@encrypted.bin" -F "key=@upb.key" --output decrypted.pdf

3. Implementácia úloh 5-6

V úlohách 5-6 sa zameriavame na generovanie a overenie digitálneho podpisu, ktoré slúžia na zachovanie pravosti a integrity dát. Pri vytváraní digitálneho podpisu (napr podpis súboru) ide o kombináciu súboru a privátneho kľúča odosielateľa. Odosielateľ

teda nešifruje súbor verejným kľúčom prijímateľa ako v úlohe 3, ale vytvorí podpis k súboru svojím vlastným privátnym kľúčom. Prijímateľ overí digitálny podpis verejným kľúčom odosielateľa. Týmto teda nedochádza k šifrovaniu dát, ale k overeniu, či súbor skutočne pochádza od pôvodného prijímateľa. Ak by man in the middle odchytil súbor odoslaný a podpísaný odosielateľom (jeho privátnym kľúčom), modifikoval ho a poslal ďalej, prijímateľ by videl, že bol súbor modifikovaný, lebo by overenie digitálneho podpisu pomocou verejného kľúča skutočného odosielateľa nebolo platné (za predpokladu, že man in the middle nemá prístup k privátnemu kľúču odosielateľa). Podobne ako v taskoch 2-4, aj tu využívame pri vytváraní/overovaní podpisu padding, konkrétne padding.PSS s parametrom MGF1. Súčasne je taktiež použité aj hashovanie.

- /api/sign ['POST'] je request na vytvorenie digitálneho podpisu – podpísanie súboru aktuálnym odosielateľom. Vstupom je súbor, ktorý sa má podpísať, v našom prípade zadanie2 v pdf formáte a privátny kľúč upb.key. Response je v podobe binárneého súboru signature .bin, ktorý obsahuje digitálny podpis.
    - curl.exe -X POST http://127.0.0.1:1337/api/sign -F "file=@zadanie2.pdf" -F "key=@upb.key" --output signature.bin
- /api/verify/<user> ['POST'] je request, kde na základe mena odosielateľa <user> je prijímateľ schopný overiť digitálny podpis. Vstupom je prijatý súbor, ktorý bol podpísaný digitálnym podpisom uloženým v súbore signature.bin v binárnom formáte. Na základe verejného kľúča sa overí pravosť digitálneho podpisu a vráti response v podobe outputu do konzoly v json formáte (true/false).
    - curl.exe -X POST http://127.0.0.1:1337/api/verify/upb -F "file=@zadanie2.pdf" -F "signature=@ signature.bin"

4. Implementácia úlohy 7

V úlohe 7 prakticky implementujeme to isté, ako v úlohe 3 a 4 na šifrovanie a dešifrovanie súboru s pridaným overením integrity súboru. Zabezpečenie integrity súboru funguje tak, že odosielateľ vytvorí hash z odosielaného (zašifrovaného) súboru, ktorý po obdržaní prijímateľom musí vygenerovať rovnaký hash, čo zaručuje, že súbor nebol modifikovaný (tampering). Mechanizmus overenia integrity súboru vykonávame pomocou hashovacej funkcie HMAC, ktorá vytvorí hash kombináciou symetrického kľúča, iv a zašifrovaného súboru. Rovnako tak pomocou HMAC funkcie vykonáme overenie hashu, teda si najskôr dešifrujeme symetrický kľúč pomocou private kľúča prijímateľa súboru, vypočítame hash zo zašifrovaného súboru s použitím dšifrovaného symetrického kľúča a iv. Následne hash porovnáme s obdržaným hashom vygenerovaným odosielateľom súboru. Ak nebol hash zmenený, pokračujeme v dešifrovaní obsahu šifrovaného súboru, ak hash overenie zlyhalo, obsah súboru nedešifrujeme

- /api/encrypt2/<user> ['POST'] je request na zašifrovanie súboru pre používateľa <user>, ktorý je prijímateľom tohto súboru. Vstupom je súbor, ktorý sa má zašifrovať v našom prípade zadanie2 v pdf formáte. Najskôr vygenerujeme symetrický kľúč pomocou AES, následne pomocou HMAC vytvoríme hash zo zašifrovaného súboru s využitím symetrického kľúča a iv a nakoniec zašifrujeme symetrický kľúč prijímateľovým verejným kľúčom. Response je v podobe balíku, ktorý obsahuje zašifrovaný symetrický kľúč, inicializačný vektor, vygenerovaný hash a zašifrovaný súbor encrypted.bin.
    - curl.exe --% -X POST 127.0.0.1:1337/api/encrypt2/upb -H "Content-Type: application/octet-stream" --data-binary @zadanie2.pdf --output encrypted_file.bin

- /api/decrypt2 ['POST'] je request na dešifrovanie prijatého súboru so zachovaním integrity. Na vstupe je zašifrovaný súbor a súbor s privátnym kľúčom prijímateľa. Najskôr pomocou privátneho kľúča prijímateľa, pre ktorého bol súbor určený/šifrovaný dešifrujeme symetrický kľúč. Po dešiforvaní symetrického kľúča pomocou HMAC overíme, či sa vypočítaný hash zhoduje s prijatým hashom v odoslanom balíku. Ak sa hash zhoduje, dešifrujeme zvyšok obsahu súboru, ak sa nezhoduje, dešifrovanie prerušíme. Response je v podobe dešifrovaného decrypted_file.pdf súboru, ktorý je totožný s pôvodným vstupným súborom zadanie2.pdf. Súčasťou response je aj správa v json formáte o (ne)úspešnosti overenia digitálneho podpisu.
    - curl.exe --% -X POST 127.0.0.1:1337/api/decrypt2 -F "file=@encrypted_file.bin" -F "key=@upb.key" --output decrypted_file.pdf