Dokumentácia k zadaniu 2

Rudolf Bezák, Radovan Križanovský

1. Výber kryptografickej knižnice pre jazyk Python

Pre riešenie tohto zadania sme vybrali kryptografickú knižnicu PyNaCl. PyNaCl je viazaná na NaCl (Networking and Cryptography Library), ktorá sa zameriava na jednoduchosť, vysokú úroveň bezpečnosti a je uznávaná vo svete kryptografie.

Inštalácia:

Knižnicu PyNaCl je možné nainštalovať pomocou príkazu:
pip install pynacl

Dôvod výberu:

PyNaCl je implementovaná na základe NaCl, ktorá je bezpečná, efektívna a využívaná na riešenie problémov s kryptografiou v praxi. Poskytuje široké spektrum funkcií vrátane asymetrickej a symetrickej kryptografie, digitálnych podpisov a autentifikácie. Zároveň je optimalizovaná pre rýchlosť a poskytuje vysoko bezpečné predvolené nastavenia, čo je pre naše zadanie ideálne.

2. Zdôvodnenie výberu asymetrickej šifry pre API volanie /api/gen/<user>

Pre generovanie asymetrických kľúčov používame algoritmus ed25519. Tento algoritmus je moderný a poskytuje vysokú úroveň bezpečnosti pri rýchlej výpočtovej náročnosti.

Dôvod výberu:

ed25519 poskytuje rýchlu generáciu verejných a privátnych kľúčov. Algoritmus je založený na eliptických krivkách, čo zaručuje vyššiu efektívnosť a menšiu veľkosť kľúčov oproti iným asymetrickým algoritmom, ako napríklad RSA. Vďaka vysokej rýchlosti je vhodný pre aplikácie, ktoré potrebujú generovať kľúče často, čo je výhodné v kontexte webových aplikácií.

3. Zdôvodnenie a popis symetrickej šifry pre API volanie /api/encrypt/<user>

Pri šifrovaní používame kombináciu symetrickej a asymetrickej kryptografie. Symetrická šifra, ktorú sme použili, je súčasťou mechanizmu SealedBox z PyNaCl, ktorý využíva šifrovací algoritmus XChaCha20-Poly1305.

Dôvod výberu:

XChaCha20-Poly1305 je symetrický šifrovací algoritmus založený na šifre ChaCha20, ktorý poskytuje vysokú bezpečnosť a zároveň rýchlosť pri spracovaní veľkých súborov. Tento algoritmus poskytuje nielen šifrovanie, ale aj autentifikáciu pomocou Poly1305, čo zaručuje ochranu integrity šifrovaných dát.

Popis formátu zašifrovaného súboru:

Formát zašifrovaného súboru je binárny. Obsahuje zašifrovaný obsah súboru a šifrovaný symetrický kľúč, pričom tento formát vychádza zo špecifikácie SealedBox, kde sa šifruje symetrický kľúč a následne aj samotný súbor pomocou algoritmu XChaCha20.

4. Zdôvodnenie mechanizmu ochrany integrity pre API volania /api/encrypt2/<user> a /api/decrypt2

Pre ochranu integrity sme zvolili mechanizmus HMAC (Hash-based Message Authentication Code). HMAC je štandardne používaný na overenie integrity a autenticity dát pomocou kryptografickej hašovacej funkcie a tajného kľúča.

Dôvod výberu:

HMAC je známy svojou odolnosťou voči kolíziám a je často využívaný v kryptografických systémoch na zabezpečenie integrity prenášaných dát. V kombinácii s algoritmom SHA-256, ktorý sa používa ako hašovacia funkcia, poskytuje vysokú úroveň bezpečnosti a robustnosť voči útokom.

Popis mechanizmu:

Pri šifrovaní sa najskôr vytvorí HMAC pomocou zdieľaného tajného kľúča a súboru, ktorý sa má zašifrovať. HMAC sa pripojí k šifrovanému obsahu. Pri dešifrovaní sa HMAC oddelí a porovná sa s HMAC vygenerovaným z dešifrovaného súboru. Ak sa HMAC kódy zhodujú, integrita súboru je zachovaná. Ak sa kódy nezhodujú, dešifrovanie sa zastaví a hlási sa chyba integrity.

5. Endpointy a ich funkcia

/api/gen/<user> (GET)

Tento endpoint generuje asymetrický kľúčový pár pre používateľa `user`. Verejný kľúč sa uloží do databázy, zatiaľ čo privátny kľúč sa vráti klientovi v binárnom formáte.

/api/encrypt/<user> (POST)

Endpoint slúži na zašifrovanie súboru pre používateľa `user`. Server vygeneruje náhodný symetrický kľúč, ktorým zašifruje obsah súboru. Tento kľúč sa následne zašifruje verejným kľúčom používateľa a spolu s obsahom sa vráti klientovi.

/api/decrypt (POST)
Endpoint dešifruje súbor, ktorý bol predtým zašifrovaný pomocou `/api/encrypt`. Klient musí poskytnúť privátny kľúč. Súbor sa dešifruje a vráti v binárnom formáte.

/api/sign (POST)
Tento endpoint vytvára digitálny podpis pre zadaný súbor pomocou privátneho kľúča klienta. Výsledný podpis sa vráti v binárnom formáte.

/api/verify/<user> (POST)
Slúži na overenie digitálneho podpisu. Verejný kľúč používateľa sa načíta z databázy a overí sa pravosť podpisu na základe verejného kľúča a podpisu.

/api/encrypt2/<user> (POST)
Rozšírená verzia šifrovania s kontrolou integrity pomocou HMAC. Súbor je zašifrovaný, k nemu je pripojený HMAC a celý výsledok je odoslaný klientovi.

/api/decrypt2 (POST)
Rozšírená verzia dešifrovania s kontrolou integrity. Po dešifrovaní sa HMAC overí, a ak kontrola integrity zlyhá, dešifrovanie sa preruší.

Literatúra:

1. curl. https://curl.se/.
2. Flask. https://flask.palletsprojects.com/en/3.0.x/.
3. Flask-sqlalchemy. https://flask-sqlalchemy.readthedocs.io/en/3.1.

Použité zdroje:

1.PyNaCl Documentation: https://pynacl.readthedocs.io

2. ed25519 Algorithm: https://ed25519.cr.yp.to

3. XChaCha20-Poly1305: https://tools.ietf.org/html/draft-arciszewski-xchacha-00

4. HMAC Mechanism: https://datatracker.ietf.org/doc/html/rfc2104.