BEADANDÓ

2019.01.21

OTP ÉS RSA

1. A project felépítése

A projektet Python 3.6.3-as verzióban írtam. Github-ot használtam a verziókövetés érdekében, aminek a linkjét a bekezdés végén megtalál. A feladat leírásnak megfelelően strukturáltam mappákba rendezve. A következő mappa nevekkel: I, II, III. Az ezekben található „test" mappa a kód gyors teszteléséért lett létrehozva a benne lévő kódokkal együtt. A főkönyvtárban az „rsc" mappában találhatóak a teszt fájlok.

A következő részeket tartalmazza a „III" mappa:

- *otp_encryption.py: az OTP osztály megvalósítása, ami tartalmazza ezen titkosításhoz szükséges függvényeket.*
- *otp_test.py: az otp_encryption.py gyors tesztelése.*
- *otp_model.py: a GUI és az OTP osztály közötti kommunikáció részét képezi.*
- *rsa_encryption.py: az RSA-hoz szükséges matematika és a titkosítás függvényeit tartalmazza.*
- *rsa_test.py: az rsa_encryption.py gyors teszteléséhez lett létrehozva.*
- *rsa_model.py: a GUI és az RSA osztály közötti kommunikáció részét képezi.*
- *origin_model.py: a modellek szülőjét, azok közös tulajdonságait és függvényeit gyűjti össze.*
- *file_handler.py: a fájlok beolvasását és kiírásához szükséges metódusokat tartalmazza.*
- *application.py: A grafikus felület.*

Megjegyzés: a kód egyes részeihez adtam angol leírásokat.

Link: https://github.com/Zarrev/InfromationAndCodingTheoryProject/

2. Lépésről lépésre – OTP, RSA

A program futtatását követően egy ablakot látunk magunk előtt 2 füllel, 2 rádió gombbal, 6 gombbal és a másik fülön még két input mezőt is találunk. A füleken tudjuk kiválasztani az éppen használni kívánt kriptográfiai műveletet. A rádiógombokkal pedig válthatunk a kódolás, illetve a dekódolás között.

OTP - Encryption

1. Bizonyosodjunk meg, hogy az OTP fülön vagyunk, valamint arról, hogy az „Encryption" módban vagyunk.
2. Töltsük be a titkosítani kívánt fájlunkat az „Open your input file" gombbal. Ez egy felugró ablak segítségével választhatjuk ki.
3. Generáljuk le és mentsük le a kulcsunkat a „Generate key and save" gombbal. A fukciót úgy írtam meg, hogy egyből be is tölti a kulcsunkat a rendszer a háttérben, így a „Load your key" gombot itt nem muszáj használni.
4. Nyomjuk meg a „Start" gombot, amivel megtörténik a háttérben a titkosítás az üzenetre nézve.
5. Mentsük el a kimenetet a „Save your output file" gombbal.
6. Lépjünk ki a programból a „Quit" gombbal.

OTP – Decryption

1. Bizonyosodjunk meg, hogy az OTP fülön vagyunk, valamint váltsunk át a „Decryption” módra, ha eddig nem ott voltunk.
2. Töltsük be a dekódolni kívánt fájlunkat az „Open your input file” gombbal. Ezután egy felugró ablak segítségével választhatjuk ki azt.
3. Töltsük be a korábban lementett kulcsunkat a „Load your key” gomb segítségével.
4. Menjünk rá a „Start” gombra, ami elvégzi a műveletet a háttérben.
5. Kattintsunk a „Save your output file” gombra és válasszuk ki a fájl helyét, ahova menteni szeretnénk, mint korábban a kulcs mentésénél.

RSA – Encryption

1. Bizonyosodjunk meg, hogy az RSA fülön vagyunk, valamint arról, hogy az „Encryption” módban vagyunk.
2. Töltsük be a titkosítani kívánt fájlunkat az „Open your input file” gombbal. Ezt egy felugró ablak segítségével választhatjuk ki.
3. Adjunk meg két prím számot az input mezőkbe (First/Second prime number).
4. Generáljuk le és mentsük le a kulcsunkat a „Generate key and save” gombbal.
5. (Jelen esetben magunknak küldünk üzenetet) A „Load your key” gombbal töltsük be a nyílvános kulcsot (public_key.txt).
6. Nyomjuk meg a „Start” gombot, amivel megtörténik a háttérben a titkosítás.
7. Mentsük el a kimenetet a „Save your output file” gombbal.
8. Lépjünk ki a programból a „Quit” gombbal.

RSA – Decryption

1. Bizonyosodjunk meg, hogy az RSA fülön vagyunk, valamint arról, hogy az „Decryption” módban vagyunk.
2. Töltsük be a dekódolni kívánt fájlunkat az „Open your input file” gombbal. Ezt egy felugró ablak segítségével választhatjuk ki.
3. A „Load your key” gombbal töltsük be a privát kulcsot (private_key.txt).
4. Nyomjuk meg a „Start” gombot, amivel megtörténik a háttérben a dekódolás.
5. Mentsük el a kimenetet a „Save your output file” gombbal.
6. Lépjünk ki a programból a „Quit” gombbal.

3. Implementáció

rsa_encryption.py

Az RSA osztályt valósítja meg, a hozzá tartozó matematikai algoritmusokkal.
A szorosan nem kapcsolodó függvényeket statikus függvényként deklaráltam. Ezek a következők:

1. gcd(a, b): az Euklideszi algoritmust használja a legnagyobb közös osztó megtalálásához.
2. extended_gcd(a, b): a kibővített Euklideszi algoritmus megvalósítása. Ezzel megkapjuk az lnko-t és a multiplikatív inverzeket a-ra és b-re nézve mod b és mod a esetekben.
3. mod_inv(a, m): habár 2. pontban leírt függvény megadja a multiplikatív inverzeket, azok lehet, hogy negatív számok, így ezt a „hibát” javítani kell. Ezzel a függvénnyel oldottam meg.
4. euler_operator(first_prime, second_prime): a $\Phi(m) = (p-1)*p(q-2)$ és a $m=p*q$ meghatározása
5. choose_e(phi_m): a megfelelő e szám kiválasztása. Néhány helyen láttam (pl.: Győri Vajda könyv), hogy megengedik az 1-et is mint értéket, de ezt nem tartottam észszerűnek ezért 2-től indítottam az intervallumot.
6. generate_keys(self, first_prime, second_prime): a fenti függvények használatával legenerálom a saját publikus és privát kulcsomat, amit egy listában (tömbben) adok vissza.

7. load_public_key(self, public_key) és load_private_key(self, private_key): a két függvény a publikus illetve a privát kulcsot betölti az adott példányra. A paraméterek tuple-ök, amik tartalmazzák a kulcsok definíció szerinti alakját. (k_p={e, m}; k_s={p,q,d})
8. encrypt(self, input_data): az input_data biteket tartalmazó string szóközzel elválasztva. Ezt listává alakítom a szóközök mentén, majd a titkosítás képletének megfelelően kódolom az üzenetet. Visszatérési érték egy szóközzel elválasztott string.
9. decrypt(self, input_data): az input_data titkosított üzenet string szóközzel elválasztva. Ezt listává alakítom a szóközök mentén, majd a dekódolás képletének megfelelően megfejtem az üzenetet. Visszatérési érték az eredeti bitek stringben szóközzel elválasztva.

otp_encryption.py

Az OTP osztályt valósítja meg. Hexába váltott karakterekkel dolgozik, hogy az utf-8 karakterkódolással is működjön.

1. abc (statikus osztály változó): a lehetséges értékek hexadecimálisan
2. load_key(self, key): betölti a kulcsot a példányra
3. gen_key(self, msg): kulcsot generál az üzenet hossza alapján. os.urandom függvény van használva, mert ez valódi véletlenszerű érték.
4. en_de_crypt(self, msg, en=True): a titkosítást és a visszafejtést valósítja meg. A két mód között az en paraméter állításával tudunk váltani. Normál állapotban titkosít az en=True default érték miatt.

file_handler.py

A fájkezelést valósítja meg az osztály, egy filename globális változóval, ami a fájl elnevezéshez van felhasználva. A tkinter standard könyvtárat használom, a pop-up ablakok használatához, hogy ezek segítségével tudjunk fájlt, illetve elérési útvonalat kiválasztani. Az os standard könyvtárat, arra használom, hogy a pop-up ablak a futtatás helyétől induljon.

1. get_file(title="Select a file for the operation!", rsa=False): fájl beolvasást valósít meg. Az rsa paraméter állításával adjuk meg, hogy hexadecimálisban adja-e vissza az értéket vagy egy stringben, ami biteket tartalmaz szóközzel elválasztva.
2. save_encrypted_file(data, new_filename=None, title="Select a directory for your file!", rsa=False): a fájl lementését valósítja meg.
3. save_key(key, name_of_file): az OTP kulcsot menti le.
4. load_key(_tuple=False): kulcsot tudunk vele betölteni. a _tuple paraméter az rsa kulcs beolvasása esetén True.
5. save_keys(keys): az RSA kulcsokat menti le.

origin_model.py

A Model osztályt valósítja meg, ami a közös tulajdonságokat és funkciókat gyűjti össze a későbbi implementációk megkönnyítésére.

otp_model.py

Az OTP_M osztályt valósítja meg ami, a GUI és az OTP működését köti össze. A Model osztályból örököl.

rsa_model.py

Az RSA_M osztályt valósítja meg ami, a GUI és az OTP működését köti össze. A Model osztályból örököl.

application.py

A GUI és annak működését valósítja meg. Egy ablakot hozok létre, amit felosztok 2 lapra a titkosítási eljárás típusának megfelelően. Függvények és osztály(ok):

1. IntegerEntry osztály: ami az input mező egy olyan megvalósítása, ami csak számokat engedélyez beírni.
2. open_btn_click(): egy esemény függvény, ami az OTP/RSA módtól függően beolvas egy fájlt, amit kiválasztunk

3. save_btn_click(): egy esemény föggvény, ami az OTP/RSA módtól függően lementi a kimenetünket, amit generáltunk a program futása során.
4. generate_key(): egy esemény függvény, ami az OTP/RSA módtól függően legenerálja és lementi a kulcsunkat/kulcsainkat a kiválasztott helyre. Az OTP esetében be is tölti egyből a kulcsot.
5. load_key(): egy esemény függvény, ami az OTP/RSA módtól függően betölti a megfelelő módon kiválasztott kulcsunkat.
6. start():egy esemény függvény, ami az OTP/RSA módtól függően elindítja a titkosítás/dekódolás folyamatát.
7. _quit():egy esemény függvény, ami kilép a programból.

A rendszer követelmény:

- Python 3.6.3
- Windows 10