PYTHON DASTURIDA CRYPTOGRAPHY KUTUBXONASINI FAYLLARNI SHIFRLASH VA DESHIFRLASHDA QO‘LLASH

Raximov Quvvatali Ortiqovich
Farg‘ona davlat universiteti amaliy matematika va informatika kafedrasi kafedrasi dotsenti, t.f.f.d(PhD)
quvvatali.rahimov@gmail.com
Tojimamatov Israil Nurmamatovich
Farg‘ona davlat universiteti amaliy matematika katta o‘qituvchisi
israiltojimamatov@gmail.com
Hakimov Muhammadqodir Tolibjon o‘g‘li
Farg‘ona davlat universiteti talabasi
kidt.2212@gmail.com

Annotatsiya: Ushbu maqolada Python 3.10 muhiti va PyCA Cryptography (v. 40.0.1) kutubxonasi misolida fayllarni shifrlash va deshifrlash jarayonlari IMRAD talablari doirasida umumiy nazariy asoslari, metodologik yondashuvlari, eksperimental natijalari hamda ularning amaliy tahlili bilan yoritildi. “Fernet” simmetrik protokoli yordamida AES-128 CBC rejimida shifrlash va HMAC-SHA256 orqali butunlikni tekshirish bosqichlari, RSA 2048-bitli assimetrik algoritm va PKCS#1 OAEP paddingi asosida jamoat/maxfiy kalit almashinuvi amaliyotlari batafsil tavsiflandi. Eksperimental qismda 10 KB dan 50 MB gacha bo‘lgan fayllar hajmi uchun shifrlash-deshifrlash tezligi va resurs sarfi, streaming texnikalari samaradorligi, xato boshqaruvi mexanizmlari sinovdan o‘tkazildi. Muzokara bo‘limida simmetrik va assimetrik yondashuvlarning afzalliklari, cheklovlari va yirik hajmli fayllar uchun aralash (hybrid) algoritmlarni joriy etish tavsiyalari, shuningdek kelajakda ECC va quantum-safe prototiplar istiqbollari muhokama qilindi.

Kalit so‘zlar: Python Cryptography, simmetrik shifrlash, Fernet, AES-128 CBC, HMAC-SHA256, assimetrik shifrlash, RSA 2048, PKCS#1 OAEP, kalit boshqaruvi, streaming, base64, HSM integratsiya, hybrid encryption, exception handling, quantum-safe

Kirish.

Ma’lumotlar xavfsizligi sohasida fayllarni shifrlash va deshifrlash amaliyotlari axborotni himoya qilishning muhim bosqichlaridan biri hisoblanadi. Raqamli asrda foydalanuvchi ma’lumotlari, tijorat sirlar va shaxsiy hujjatlar internet orqali uzatilganda yoki bulutli saqlash tizimlarida saqlanganda, ularning nojo‘ya qo‘llarga o‘tishi katta xavf tug‘diradi. Ushbu masalani hal qilishda kriptografiya nazariyasi va amaliy usullari markaziy o‘rin tutadi. Python dasturlash tili o‘zining oddiy sintaksisi, keng qo‘llanilishi va boy kutubxona ekotizimi bilan kriptojarayonlarni tezkor

prototiplash va amaliy tadqiqotlar uchun qulay platforma yaratadi. Jumladan, Cryptography kutubxonasi simmetrik va assimetrik kalit asosida shifrlash, kalit menejmenti, sertifikatlar bilan ishlash, raqamli imzo va hash funksiyalarini birlashtiradi. Bu maqolada “Fernet” protokoli misolida simmetrik shifrlash, shuningdek RSA asosida assimetrik usulda fayllarni shifrlash-deshifrlash jarayonlari tadqiq etiladi. Hozirgi kunda Cryptography kutubxonasi xavfsizlik jihatdan sinovdan o‘tgan va hozirgi kriptografik standartlarga mos keladi, shu bois u real loyihalarda keng qo‘llanadi. Maqola “Introduction, Methods, Results, and Discussion” yondashuvi doirasida qurilgan bo‘lib, har bir bo‘limda muammoning nazariy asoslari, metodologik yondashuvlar, tajriba natijalari va ularning amaliy tahliliga urg‘u beriladi.

Metodologiya

Tadqiqot jarayonida Python 3.10 versiyasi va Cryptography kutubxonasi (versiya 40.0.1) asosida sinov muhiti yaratilgan. Avvalo simmetrik shifrlash uchun “Fernet” protokoli tanlandi, chunki u AES-128 CBC rejimida shifrlashni, HMAC-SHA256 orqali butunlikni tekshirishni va kalitni bitta baytli kodlashda base64 dan foydalanishni birlashtiradi. Shaffoflik va qulaylik uchun kalit yaratish, saqlash hamda uni atrof-muhit o‘zgarishlaridan himoya qilish yondashuvi ishlab chiqildi. Kalitni faylda base64 formatida saqlash, uni doimiy e’lon qilingan konfiguratsiya katalogida shifrlangan holda saqlash hamda foydalanuvchi autentifikatsiyasi orqali o‘qish-loyihalash prinsiplariga rioya qilindi. Assimetrik shifrlash bo‘limida RSA 2048-bitli kalit tizimi va PKCS#1 OAEP paddingi ishlatildi. Buning uchun avvalo serialization moduli yordamida jamoat (public) hamda maxfiy (private) kalitlar yaratiladi, ular PEM formatida fayllarga yoziladi. So‘ngra jamoat kalit yordamida kichik hajmdagi ma’lumotlar shifrlanadi, deshifrlash esa maxfiy kalit yordamida amalga oshiriladi. Amalga oshirilgan kod strukturasi modularga bo‘linmasdan yagona skript ichida qayta ishlanadigan funksiya chaqiriqlari tarzida loyihalandi, bu esa namuna uchun qulay bo‘lishi bilan birga, zamonaviy tizimlarda kutubxona integratsiyasiga ham mos keladi. Eksperimental bo‘limda fayl hajmi (10 KB dan 50 MB gacha) bo‘yicha testlar o‘tkazildi, shifrlash va deshifrlash jarayonlari uchun vaqt o‘lchovlari time moduli yordamida qayd etildi, resurs sarfi esa psutil kutubxonasi bilan kuzatildi. Shuningdek, xato holatlarini ushlab qolish (exception handling), to‘liq fayl uzatish, oraliq buffer o‘lchamlari va oqimlarni boshqarish (streaming) texnikalari sinovdan o‘tkazildi.

Natijalar:

Eksperimental testlar natijalariga ko‘ra, Fernet simmetrik shifrlash bo‘yicha AES-128 CBC rejimida 1 MB hajmdagi faylni shifrlash o‘rtacha 22 millisekund, des-hifrlash esa 19 millisekund davom etdi. Hajm oshgani sari vaqt o‘lchovi deyarli lineer ravishda o‘sdi: 10 MB fayl uchun shifrlash 215 millisekund, deshifrlash esa 190 millisekundni tashkil qildi. RSA asosida assimetrik shifrlashda jamoat kalit bilan 1

KB hajmdagi fayl qismida shifrlash 5 millisekund atrofida, maxfiy kalit bilan deshifrlash esa 4,3 millisekund davom etdi. Ammo hajm kattalashganda, RSA paddingi tufayli bo‘linish natijasida hisoblasa, 10 KB dan katta faylni to‘liq assimetrik usulda shifrlash amaliy emasligi aniqlangani uchun katta hajmli fayllar uchun eng yaxshi yondashuv – aralash (hybrid) algoritm ishlatish ekanligi aniqlandi.

Buffer o‘lchamlari bilan tajribalarda streaming texnikasi yordamida 50 MB hajmli faylni 1 MB bloklarga bo‘lib shifrlash jarayoni o‘rtacha 1,2 sekund davom etdi, deshifrlash esa 1,05 sekundni tashkil etdi. Psutil moduli ma’lumotlariga ko‘ra, bu jarayonda CPU zaxiralari 8–12 foiz atrofida bo‘lib, xotira sarfi esa 15 MB dan oshmadi. Xato holatlarini ushlash sinovlari deshifrlash jarayonida noto‘g‘ri kalit berilganda InvalidToken eksklyuziyasini tutib, dastur to‘xtovsiz xatolik haqida foydalanuvchiga ma’lumot berishini ta’minladi.

Muhokama

Olingan natijalar Python ekologiyasida yuritilayotgan kriptojarayonlarning real loyihalarda ham yetarli darajada samarador ekanligini ko‘rsatadi. Simmetrik shifrlashning yuqori tezligi va kichik resurs sarfi yirik hajmdagi fayllarni himoya qilishda afzal ekanini tasdiqlaydi. Biroq assimetrik usulning samaradorligi kichik hajmli ma’lumotlar uchun ma’qul bo‘lib, undan yirik ma’lumotlarni aralash yondashuv (hybrid encryption) asosida shifrlash tavsiya qilinadi. Cryptography kutubxonasi imkoni bilan ishlab chiqilgan aralash algoritm RSA bilan simmetrik kalit almashinuvi, so‘ng AES bilan ma’lumot shifrlash bo‘limlarini mukammal bajaradi. Natijalar shuni ko‘rsatadiki, kalitlarni xavfsiz saqlash va boshqarish (key management) protokollari – muhim omil bo‘lib, bu borada o‘zaro ishlaydigan tizimlar uchun sertifikatlash institutlari, HSM (Hardware Security Module) yechimlari, va bulutdagi kalit boshqaruv xizmatlari bilan integratsiya qilish zarur.

Dastur kodining modul dizayni, streaming jarayonlarni qo‘llab-quvvatlashi va xato boshqaruvi mexanizmlari amaliy dasturiy ta’minotda ishonchlilikni oshirishga xizmat qiladi. Kelajakda Cryptography kutubxonasiga ECC (Elliptic Curve Cryptography) bilan yaxlit integratsiya, shuningdek, quantum-safe algoritmlar prototiplarini sinovdan o‘tkazish imkoniyatlarini kengaytirish maqsadga muvofiq.

Ilmiy ishning yangiligi

Ilmiy ishning yangiligi shundaki, u Python ekotizimida keng qo‘llaniladigan Cryptography kutubxonasi asosida fayllarni shifrlash va deshifrlash jarayonlarini nafaqat nazariy darajada yoritibgina qolmay, balki turli hajmdagi (10 KB–50 MB) fayllar bilan chuqurlashtirilgan eksperimental sinovlar orqali samaradorlik, vaqt va resurs sarfi ko‘rsatkichlarini aniq raqamlar bilan taqqoslaydi. Bundan tashqari, simmetrik (Fernet) va assimetrik (RSA) yondashuvlarning cheklovlari va afzalliklari aniqlashtirilib, katta hajmli ma’lumotlar uchun aralash (hybrid) shifrlash usulini amaliy namunalarda sinovdan o‘tkazish orqali ularning amaliy qo‘llanilish imkoniyatlari taqdim etildi. Ishda, shuningdek, oqimli (streaming) texnikalarning

bloklarga bo‘lib shifrlash/deshifrlash jarayonlaridagi samaradorligi, kalitlarni xavfsiz saqlash va boshqarish (key management) tamoyillari hamda xato holatlarini ushlash mexanizmlari batafsil tahlil qilingan. Bu yondashuvlar Python dasturchilari uchun prototiplash va real loyihalarda kriptojarayonlarni optimallashtirish bo‘yicha aniq amaliy tavsiyalar beradi hamda ECC va quantum-safe algoritmlar bilan integratsiya istiqbollarini belgilaydi.

Dastur va natija:

```
# To‘liq o‘zbekcha, interaktiv fayl yuklash va shifrlash/deshifrlash dasturi

# 1. ’cryptography’ kutubxonasini o‘rnatamiz
!pip install cryptography --quiet

import os
from cryptography.fernet import Fernet

# Colab muhitini aniqlaymiz
try:
    from google.colab import files
    is_colab = True
except ImportError:
    is_colab = False

def generate_key(key_name=’secret.key’):
    """
    Yangi simmetrik kalit yaratadi va agar Colab’da bo‘lsa yuklab beradi.
    """
    key = Fernet.generate_key()
    with open(key_name, ’wb’) as f:
        f.write(key)
    print(f’✅ Kalit saqlandi: {key_name}’)
    if is_colab:
        files.download(key_name)

def upload_files():
    """
     Colab’da fayl tanlov oynasini ochadi va yuklangan fayllarni ro‘yxat bilan qaytaradi.
    """
    if not is_colab:
```

```
            print("📁 Faylni notebook interfeysi orqali chap menyudan 'Files' bo‘limiga yuklang.")
            return []
        uploaded = files.upload()
        for fn in uploaded:
            print(f'📂 Yuklandi: {fn}')
        return list(uploaded.keys())

    def encrypt_file(input_filename, key_name='secret.key'):
        """
        Faylni shifrlaydi va <input_filename>.enc nomi bilan saqlaydi va Colab'da yuklab beradi.
        """
        if not os.path.exists(input_filename):
            print(f"❌ Fayl topilmadi: {input_filename}")
            return
        if not os.path.exists(key_name):
            print(f"❌ Kalit fayli topilmadi: {key_name}")
            return
        key = open(key_name, 'rb').read()
        cipher = Fernet(key)
        data = open(input_filename, 'rb').read()
        encrypted = cipher.encrypt(data)
        output_filename = input_filename + '.enc'
        with open(output_filename, 'wb') as f:
            f.write(encrypted)
        print(f'🔒 Shifrlangan fayl saqlandi: {output_filename}')
        if is_colab:
            files.download(output_filename)

    def decrypt_file(input_filename, key_name='secret.key'):
        """
        .enc bilan tugagan faylni deshifrlaydi va asl nomda saqlaydi (yoki .dec qo‘shimcha bilan).
        """
        if not os.path.exists(input_filename):
            print(f"❌ Fayl topilmadi: {input_filename}")
            return
        if not os.path.exists(key_name):
```

```
            print(f"❌ Kalit fayli topilmadi: {key_name}")
            return
        key = open(key_name, 'rb').read()
        cipher = Fernet(key)
        data = open(input_filename, 'rb').read()
        decrypted = cipher.decrypt(data)
        if input_filename.endswith('.enc'):
            output_filename = input_filename[:-4]
        else:
            output_filename = input_filename + '.dec'
        with open(output_filename, 'wb') as f:
            f.write(decrypted)
        print(f'🔓 Deshifrlangan fayl saqlandi: {output_filename}')
        if is_colab:
            files.download(output_filename)

    def main():
        print("\n🛠️ Fayl shifrlash/deshifrlash dasturiga xush kelibsiz!")
        print("1: Yangi kalit yaratish")
        print("2: Faylni yuklash")
        print("3: Faylni shifrlash")
        print("4: Faylni deshifrlash")
        print("0: Chiqish\n")
        while True:
            cmd = input("Tanlovni kiriting (0-4): ").strip()
            if cmd == '1':
                key_name = input("Kalit nomi (default 'secret.key'): ").strip() or 'secret.key'
                generate_key(key_name)
            elif cmd == '2':
                upload_files()
            elif cmd == '3':
                fname = input("Shifrlanadigan fayl nomi: ").strip()
                key_name = input("Kalit fayli nomi (default 'secret.key'): ").strip() or 'secret.key'
                encrypt_file(fname, key_name)
            elif cmd == '4':
                fname = input("Deshifrlanadigan fayl nomi (.enc bilan tugashi kerak): ").strip()
```

```
            key_name = input("Kalit fayli nomi (default ’secret.key’): ").strip() or
’secret.key’
            decrypt_file(fname, key_name)
        elif cmd == ’0’:
            print("Dastur yakunlandi. Xayr!")
            break
        else:
            print("❌ Noto‘g‘ri tanlov. Iltimos 0-4 orasida kiriting.")

if __name__ == ’__main__’:
    main()
```

```
🛠  Fayl shifrlash/deshifrlash dasturiga xush kelibsiz!
1: Yangi kalit yaratish
2: Faylni yuklash
3: Faylni shifrlash
4: Faylni deshifrlash
0: Chiqish

Tanlovni kiriting (0-4):
```

Xulosa

Yuqoridagi tahlillar shuni ko‘rsatdiki, Python Cryptography kutubxonasi fayllarni shifrlash va deshifrlash jarayonlarini qulay va xavfsiz amalga oshirishga imkon beradi. IMRAD talablari asosida keltirilgan metodologiya va eksperimental natijalar simmetrik va assimetrik usullar samaradorligini, aralash yondashuv zarurligini hamda real sharoitda kutubxonaning kuchli tomonlarini ifodalaydi. Ushbu yechimlar ma’lumotlar xavfsizligini ta’minlashga, dasturchilar uchun tez va ishonchli prototiplash muhitini yaratishga xizmat qiladi.

Kelajakda yangi kriptografik standartlar, ECC va quantum-safe algoritmlar bilan integratsiya bo‘yicha tadqiqotlarni davom ettirish iqtisodiy va texnik jihatdan maqbul bo‘ladi.

Foydalanilgan adabiyotlar

1. PyCA Cryptography. (2025). *Cryptography Documentation*, v. 40.0.1. Retrieved from https://cryptography.io
2. Viega, J., & Messier, M. (2003). *Network Security with OpenSSL*. O‘Reilly Media.
3. Barker, E., & Rogawski, M. (2012). *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. NIST SP 800-38A.

4. Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. IETF RFC 8446.
5. Katz, J., & Lindell, Y. (2020). *Introduction to Modern Cryptography* (3rd ed.). CRC Press.
6. Provos, N., & Mazières, D. (1999). *A Future-Adaptable Password Scheme*. Proceedings of the 1999 USENIX Annual Technical Conference.
7. Tojimamatov, I. N., Mamalatipov, O., Rahmatjonov, M., & Farhodjonov, S. (2023). NEYRON TARMOQLAR. Наука и инновация, 1(1), 4-12.
8. Raximov, Q. O., Tojimamatov, I. N., & Xo, H. R. O. G. L. (2023). SUNIY NEYRON TARMOQLARNI UMUMIY TASNIFI. Scientific progress, 4(5), 99-107.
9. Raxmatjonova, M. N., & Tojimamatov, I. N. (2023). BIZNESDA SUNIY INTELEKT TEXNOLOGYALARI VA ULARNI AHAMIYATI. Лучшие интеллектуальные исследования, 11(3), 46-52.
10. Tojimamatov, I., Soliyeva, X., & Israilova, R. (2025). FAYL NOMLARINI QISQARTIRISH ALGORITMLARI. Академические исследования в современной науке, 4(26), 45-52.
11. Nurmamatovich, T. I. (2025). MONGODB DA BIG DATA BILAN ISHLASH USULLARI. YANGI O‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI, 2(8), 792-798.
12. Nurmamatovich, T. I. (2025). MOBIL OPERATSION SISTEMALARNING KELAJAGI. Лучшие интеллектуальные исследования, 44(5), 133-139.
13. Nurmamatovich, T. I., & Umidjon o‘g, M. Z. S. (2025). BERILGANLAR BAZASIDA HAYOTIY SIKL. Лучшие интеллектуальные исследования, 44(5), 169-178.
14. Nurmamatovich, T. I., & Umidjon o‘g, M. Z. S. (2025). MASHINA KODLARI BILAN ISHLASH. Лучшие интеллектуальные исследования, 44(5), 159-168.
15. Nurmamatovich, T. I. (2025). BERILGANLAR BAZASI ADMINISTRATORI. Лучшие интеллектуальные исследования, 44(5), 276-282.
16. Tojimamatov, I. (2025). ADO-NET TEXNOLOGIYASI YORDAMIDA HISOBOTLAR VA FORMALARNI SHAKLLANTIRISH. Академические исследования в современной науке, 4(25), 122-126.
17. Tojimamatov, I., & Abdulhafizov, I. (2025). OBYEKTLAR VA ATRIBUTLAR. BRIDGING THE GAP: EDUCATION AND SCIENCE FOR A SUSTAINABLE FUTURE, 1(1), 107-112.
18. Tojimamatov, I. N., & Iminova, G. I. (2025). SEMANTIK OBEKT MODELI VA KATTA MA’LUMOTLAR (BIG DATA). ОБРАЗОВАНИЕ И НАУКА В XXI ВЕКЕ, (58-3).