UOT 004.42

DOI: 10.25045/ASUCAAI.2025.28

PYTHON KİTABXANALARINDAN İSTİFADƏ EDƏRƏK ADVANCED ENCRYPTION STANDARD (AES) TƏTBİQİ

İlham Cəbrayılov Qardaşxan oğlu
Dosent
Azərbaycan Texniki Universiteti
Bakı, Azərbaycan
ilham.cebrayilov@aztu.edu.az
Aydan Hümmətova Səməd qızı
Baş müəllim
Azərbaycan Texniki Universiteti
Bakı, Azərbaycan
aydan.hommatova@aztu.edu.az

Xülasə

Məqalədə AES şifrələməsinin əsas tətbiqini təmin edən və təhlükəsizlik praktikaları olan açar törəmədən istifadə qaydaları nəzərdən keçirilir. O, Python proqramlarında AES şifrələməsinin idarə edilməsinin əsas anlayışlarını başa düşmək üçün nümunə təqdim edir. Bu skript AES şifrələməsinin əsas anlayışlarını anlamaq istəyən tərtibatçılar və həvəskarlar üçün başlanğıc bələdçisi kimi xidmət etməklə AES şifrələməsinin prinsiplərini başa düşmək rəqəmsal dünyada məxfilik və bütövlüyü təmin etmək üçün çox vacibdir.

Python proqramlarında AES şifrələməsindən istifadə təkcə təhlükəsizlik tədbiri deyil, həm də həssas məlumatları icazəsiz girişdən qorumaq və istifadəçi məxfiliyini təmin etmək üçün vacib təcrübədir. Texnologiya inkişaf etdikcə, AES kimi möhkəm şifrələmə üsullarının bilikləri və tətbiqi həssas məlumatların qorunması üçün açardır.

Python-da AES şifrələməsinə dərindən nəzər salmaqla, tərtibatçılar tətbiqlərinin təhlükəsizliyini gücləndirmək və bununla da həssas məlumatların idarə edilməsində inamı və etibarlılığı artırmaq üçün təchiz olunurlar. Bu nəticə işin əsas məqamlarını ümumiləşdirir, Python-da AES şifrələməsinin əhəmiyyətini və məlumat təhlükəsizliyinin gücləndirilməsində əhəmiyyətini ümumiləşdirir.

Ilham Jabrailov
Aydan Hummatova

Implementing Advanced Encryption Standard (AES) Using Python Libraries

Abstract

This article discusses the rules of key derivation that provides the basic implementation of AES encryption and the security techniques. It provides an example to understand the basic concepts of AES encryption processing in Python programs. This script serves as a guide for beginner developers and enthusiasts who want to understand the basic concepts of AES encryption, as understanding the principles of AES encryption is crucial to ensure confidentiality and integrity in the digital world.

Using AES encryption in Python programs is not only a security measure but also an important practice to protect sensitive data from unauthorized access and ensure user privacy. As technology advances, knowledge and implementation of strong encryption techniques like AES become key to protecting sensitive data.

By learning AES encryption in Python in detail, developers will be able to strengthen the security of their applications, thereby increasing the trust and reliability of sensitive data handling. This conclusion summarizes the main points of the paper, generalizes the importance of AES encryption in Python and its implications for enhancing data security.

Ильхам Джабраилов
Айдан Гумматова

Реализация стандарта расширенного шифрования (AES) с использованием библиотек Python

Аннотация

В статье рассматриваются правила использования деривации ключа, обеспечивающей базовую реализацию шифрования AES, а также методы обеспечения безопасности. В нем представлен пример для понимания основных концепций обработки шифрования AES в программах на Python. Этот скрипт служит руководством для начинающих разработчиков и энтузиастов, желающих понять основные концепции шифрования AES, поскольку понимание принципов шифрования AES имеет решающее значение для обеспечения конфиденциальности и целостности в цифровом мире.

Использование шифрования AES в программах Python – это не только мера безопасности, но и важная практика защиты конфиденциальных данных от несанкционированного доступа и обеспечения конфиденциальности пользователей. По мере развития технологий знание и внедрение надежных методов шифрования, таких как AES, становятся ключевыми для защиты конфиденциальных данных.

Подробно изучив шифрование AES в Python, разработчики получат возможность усилить безопасность своих приложений, тем самым повысив доверие и надежность обработки конфиденциальных данных. В этом заключении суммируются основные положения работы, обобщается важность шифрования AES в Python и его значение для усиления безопасности данных.

Açar sözlər: Şifrələmə, təhlükəsizlik, kodlaşdırma, informasiya, simmetrik şifrələnmə.
Keywords: Encryption, Security, Coding, Information, Symmetric Encryption.
Ключевые слова: Шифрование, безопасность, кодирование, информация, симметричное шифрование.

Giriş

Müasir bir-biri ilə əlaqəli dünyada həssas məlumatların qorunması böyük narahatlıq doğurur. Şifrələmə, məlumatların oxunmaz formata çevrilməsi prosesi məlumatların təhlükəsizliyini təmin etmək üçün təməl məsələlərdən biridir. Bu sahədə iki əsas üsul, simmetrik və asimmetrik şifrələmə böyük əhəmiyyət kəsb edir. Məqalə Qabaqcıl Şifrələmə Standartına (AES), onun rejimlərinə və onların Python proqramlaşdırmasında tətbiqlərinə diqqət yetirərək bu şifrələmə üsullarını araşdırmaq məqsədi daşıyır. Bunun uçün müəlliflər tərəfindən yeni alqoritm verilmiş və Python proqlaşdırma dilində kodlaşdırılmışdır.

Simmetrik və asimmetrik şifrələnmə nədir, necə işləyir?

Simmetrik şifrələnmə dedikdə məlumatın şifrələnməsi və deşifrasiya olunması üçün eyni açardan istifadə olunur. Simmetrik şifrələnmədən istifadənın üstün tərəfləri olduğu qədər funksionallıq baxımdan çatışmayan tərəflərdə mövcuddur. Üstün tərəflərindən bəhs etsək ilk öncə prosesin sürətli olması gəlir. Bundan başqa şifrə kiminsə əlinə keçmədiyi müddətcə məlumatın tamlığını və məxfiliyini qorumaq mümkün olur. Mənfi tərəflərinə gəldikdə isə şifrələrin saxlanması

emal olunması üçün bir sistem qurmaq çox çətindir. Yəni hər bir məlumat axışı üçün fərqli açarlardan istifadə edilir. Digər mənfi tərəfi məlumatın kim tərəfdən göndərildiyini təsdiqləmək olmur.

Asimmetrik şifrələmədə isə iki ədəd açardan istifadə olunur. Public və v açarlar. Public açarla şifrələnmiş məlumat yalnız həmin public açara uyğun, private açar ilə isə deşifrasiya etmək olur. Burada əsas nüans private açardan public açar əldə etmək asan oldğu halda əksinə əməliyyatı etmək demək olar mümkünsüzdür(illərlə vaxt lazımdır). Asimmetrik şifrələnmənin müsbət tərəflərinə gəldikdə məlumatın tamlığı, məlumatın kim tərəfdən göndərildiyini təsdiqləmə və açar distribution (açar paylaşımı) mühitinin olmasıdır. Mənfi tərəfinə gəldikdə isə əməliyyatların yavaş olmasıdır.

İndi isə gəlin https qoşulma zamanı prosesin necə getdiyinə baxaq. Fərz edək ki hansısa bir saytı browser üzərindən açmağa cəhd edirik.

1. Brouzer üzərində saytın adını yazırıq DNS serverlər həmin saytın IP ünvanını bizə geri döndərir və biz həmin saytın serverinə qoşulmaq üçün müraciət edirik.
2. Həmin Web sayt bizə öz sertifikatı ilə public açarını göndərir.
3. Biz həmin sertifikatın həqiqiliyini yoxlayırıq. Serifikatın həqiqiliyini yoxlamaq üçün həmin sertifikatın hansı root CA tərəfindən imzalandığına baxırıq. Əgər həmin root CA bizim trusted root CA siyahımızda varsa buna güvənirik. Yoxdurda browserdə untrusted root CA tərəfindən imzalandığı ilə bağlı bildiriş görəcəyik.
4. Daha sonra həmin web saytın bizə göndərdiyi “public” açar ilə şifrələnmiş secret açarı-i qarşı tərəfə göndəririk. Həmin Web sayt öz public açarı ilə imzalanmış məlumatı “private” açarı ilə açır və “secret key”i əldə edir. Daha sonra bu secret key ilə təhlükəsiz kanal yaradıb məlumat axınına start verir. Sonda hər ikisində secret key olduğu üçün hər iki təfəf məlumatı deşifrasiya edə bilirlər.

Metodlar

İnformasiya təhlükəsizliyini təhdidlərdən qorumaq ücün müəxtəlif üsullar işlənib hazırlanmışdır. Məqalədə kodlaşdırma nəzəriyyəsindən və pyhton proqramlaından istifadə olunmuşdur.

Pythonda AES-nin tətbiqi: Proqram təminatı dəstəyi

Python-un cryptography kitabxanası AES şifrələməsini və onun müxtəlif rejimlərini həyata keçirmək üçün dəstək verir. Bu kitabxana AES şifrələmə əməliyyatlarını yerinə yetirmək üçün təkmilləşdirilmiş funksiyalar təklif edir və müxtəlif rejimlərdən istifadə edərək təhlükəsiz məlumat şifrələməsi üçün istifadə edilə bilər. Bu, tərtibatçılara təhlükəsizlik yönümlü proqramları inkişaf etdirmək üçün alət təqdim edir. Tədqiq olunan məsələ üçün aşagıdakı kod yazılmışdır,

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad
import hashlib
class AESCipher:
      def __init__(self, key):
            self.key = hashlib.sha256(key.encode('utf-8')).digest()
def encrypt(self, data):
      iv = get_random_bytes(AES.block_size)
      cipher = AES.new(self.key, AES.MODE_CBC, iv)
      encrypted = cipher.encrypt(pad(data.encode('utf-8'), AES.block_size))
      return iv.hex() + encrypted.hex()
def decrypt(self, data):
      iv = bytes.fromhex(data[:32]) # Assuming the IV is 16 bytes (32 hex
characters)
      ciphertext = bytes.fromhex(data[32:])
      cipher = AES.new(self.key, AES.MODE_CBC, iv)
      decrypted = unpad(cipher.decrypt(ciphertext), AES.block_size)
      return decrypted.decode('utf-8')
```

```
def encrypt_file(self, input_file, output_file):
      with open(input_file, 'rb') as file:
            plaintext = file.read()
            iv = get_random_bytes(AES.block_size)
            cipher = AES.new(self.key, AES.MODE_CBC, iv)
            encrypted = cipher.encrypt(pad(plaintext, AES.block_size))
with open(output_file, 'wb') as file:
      file.write(iv + encrypted)
      def decrypt_file(self, input_file, output_file):
with open(input_file, 'rb') as file:
      ciphertext = file.read()
      iv = ciphertext[:AES.block_size]
      cipher = AES.new(self.key, AES.MODE_CBC, iv)
      decrypted = unpad(cipher.decrypt(ciphertext[AES.block_size:]),
      AES.block_size)
with open(output_file, 'wb') as file:
      file.write(decrypted)
if __name__ == '__main__':
print('----Options----')
print('1. Encrypt/Decrypt String')
print('2. Encrypt/Decrypt File')
choice = input('Enter your choice (1 or 2): ')
password = input('Enter the password: ')
aes_cipher = AESCipher(password)
if choice == '1':
print('----Encryption----')
message = input('Enter the message you want to encrypt: ')
ciphertext = aes_cipher.encrypt(message)
print('Ciphertext:', ciphertext)
print('\n----Decryption----')
ciphertext_input = input('Enter the ciphertext (in hexadecimal format): ')
decrypted_message = aes_cipher.decrypt(ciphertext_input)
print('Decrypted Message:', decrypted_message)
elif choice == '2':
print('----File Encryption----')
input_file = input('Enter the input file name: ')
output_file = input('Enter the output file name for encrypted content: ')
aes_cipher.encrypt_file(input_file, output_file)
print('File Encrypted Successfully!')
print('\n----File Decryption----')
input_file = input('Enter the input file name for decryption: ')
output_file = input('Enter the output file name for decrypted content: ')
aes_cipher.decrypt_file(input_file, output_file)
print('File Decrypted Successfully!')
else:
print('Invalid choice. Please enter either 1 or 2.')
```

Let’s show the implementation of program using example below.
In the example below,the word “Turkan” is entered into the program to be encrypted. The Enter button is clicked, and as a result, we obtain the encrypted from of the word “Turkan”. To verify the program, we re-enter the encrypted form into the program, and this time, we see the word “Turkan” on the screen as the result. Thus, the program performs its task correctly.

Example:

```
----Decryption----
Enter the ciphertext (in hexadecimal format):
e768f54efb345aa6038187c74bc4fe7aecca62f802a8e446ab9868dc9ab88c3f
```

```
Decrypted Message: Turkan
PS C:\Users\User>
As it is seen the word Turkan is encrypted with
e768f54efb345aa6038187c74bc4fe7aecca62f802a8e446ab9868dc9ab88c3f
and decrypted.
It is possible to encode any large amount of information with the presented
algorithm, using the presented algorithm.
```

Nəticə

İnformasiya ehtiyatlarının təhlükəsizliyinin təmin edilməsi lokal və qlobal informasiya şəbəkələrində dövr edən informasiyanın hiss olunacaq dərəcədə effektivliyinin artmasına, tamlığının, həqiqiliyinin və məxviliyinin təmin edilməsinə səbəb olur. Bunun üçün müdafiyə obyektləri, informasuyanın qorunma üsulları, yayılmadan qorunması, müdafiyə vasitələri komplekslərını, onların texniki və proqram vasitələrinə və s kimi məsələlərə diqqət yetirmək lazımdır. Xüsusi halda məqalədə ənənəvi qorunma üsullarından fərqli olaraq yeni qorunma üsul təklif edilmiş və onun alqoritmi və proqram təminatı işlənmişdir.

Müzakirə: Texnologiya və onlayn qarşılıqlı əlaqəsi ilə doymuş müasir dünyamızda kibertəhlükəsizliyin əhəmiyyəti virtual məkanın hüdudlarını aşır və hər birimizə gündəlik həyatımızda təsir edir. Həyatımız həmişəkindən daha çox rəqəmsal dövrlə qarışır, burada şəxsi məlumatları paylaşır, maliyyə əməliyyatları icra edir və birbaşa cihazlarımızın ekranları vasitəsilə iş aparırıq. Bununla belə, bu rəqəmsal təkamül təkcə faydaları ilə deyil, həm də təhlükəsizliyimizi poza biləcək qlobal təhlükələri də gətirir. Ona görədə informasiya təhlükəsizliyi məsələlərinin tədqiqi həmişə aktualdır. Məqalənin nəticələri Azərbaycan Texniki Universitetinin “Mühəndis riyaziyyatı və süni intellekt” kafedrasının elmi seminarlarında müzakirə edilmişdir.

Ədəbiyyat siyahısı:

1. R.A.Susi. Python programming language. M.: “Binom. Knowledge Laboratory”, 2020.
2. M. Summerfield. Programming in Python 3. A detailed guide (translated from English) St. Petersburg: “Symbol-Plus”, 2019.
3. Björck F., Henkel M., Stirna J., Zdravkovic J. Cyber Resilience – Fundamentals for a Definition. In: Rocha A., Correia A., Costanzo S., Reis L. (eds) New Contributions in Information Systems and Technologies. Advances in Intelligent Systems and Computing, vol 353. Springer, Cham, 2015.
4. Dobrygowski, D.: Cyber resilience: everything you (really) need to know, available at http:// https://www.weforum.org/agenda/2016/07/cyb er-resilience-what-to-know/ (Accessed: 21st September 2022).
5. Cyber Defense, available at https://www.techopedia.com/definition/6705/cy ber-defense (Accessed: 10th April 2022).
6. Exclusive Networks: Unknown Unknowns – The Ultimate Test for Cybersecurity, available at http://www.exclusive-networks.com/uk/blog/unknown-unknowns-ultimate-test-cybersecurity/ (Accessed: 1st August 2022).
7. Goche, M., Gouveia, W.: Why Cyber Security Is Not Enough: You Need Cyber Resilience, available at https://www.forbes.com/sites/sungardas/2014/0 1/15/why-cyber-security-is-not-enough-you-need-cyber-resilience/#562402a21bc4 (Accessed: 1st July 2022).
9. Hulme, G.V: Security spending continues to run a step behind the threats, available at http://www.csoonline.com/article/2134074/strategic-planning-erm/security-spending- continues-to-run-a-step-behind-the- threats.html (Accessed: 13th June 2022).
10. Infosecurity, available at http://infosecurityinc.net/wp- content/uploads/2011/07/Consult-Cyber-1Cyber-Threats-Diminishing-Attack-Costs-gaIncreasing-Complexity4.jpg (Accessed: 18th September 2022).
11. Marvell, S.: The real and present threat of a cyber breach demands real-time risk management, Acuity Risk Management, 2015.
12. NATO Cyber Cooperative Cyber Defense Center of Excellence Tallin Estonia, available at https://ccdcoe.org/cyber-definitions.html (Accessed: 10th April 2022).

13. Pescatore, J.: Toward a National Cybersecurity Strategy, G00167598, Gartner, Inc., 2009.
14. Tucker, E.: Official: FBI probing attempted cyber breach of NY Times, available at http://www.federaltimes.com/articles/official-fbi-probing-attempted-cyber-breach-of-ny-times (Accessed: 31st May 2022).
15. Walls, A., Perkins, E., Weiss, J.: Definition: “Cybersecurity”, G00252816, Gartner, Inc., 2013.
16. Wheeler, J. A.: Emerging Risks in Cybersecurity: Gartner’s Top Ten Predicitons, available at http://blogs.gartner.com/john-wheeler/gartner-top-ten-cybersecurity-predicts/ (Accessed: 2nd June 2022).
17. United States Department of Defense: Strategy for Operating in Cyberspace, Department of Defense, 2011.
18. Galinec, D., Steingartner, W., Zebić, V.: Cyber Rapid Response Team: An Option within Hybrid Threats. 2019 IEEE 15th International Scientific Conference on Informatics, INFORMATICS' 2019, November 20th-22nd, 2019, Poprad, Slovakia, PROCEEDINGS, Institute of Electrical and Electronics Engineers,Inc.