Κρυπτογαραφία και Ασφάλεια

Κρυπτογράφηση χρησιμοποιώντας τον AES αλγόριθμο

Vasileios Nastos
MSc in department of Informatics
and Telecommunications
University of Ioannina
Arta, 47100, Greece

June 22, 2023

1 Εισαγωγή

- **iv =os.urandom(algorithms.AES.block_size // 8)**: Αυτή η γραμμή παράγει ένα τυχαίο διάνυσμα αρχικοποίησης (IV) χρησιμοποιώντας τη συνάρτηση os.urandom(). Το IV είναι μια τυχαία τιμή που χρησιμοποιείται για την αρχικοποίηση του αλγορίθμου κρυπτογράφησης και προσθέτει ένα επιπλέον επίπεδο τυχαιότητας στη διαδικασία κρυπτογράφησης. Το μέγεθος του IV είναι ίσο με το μέγεθος μπλοκ του αλγορίθμου AES διαιρεμένο με το 8.

- **cipher = Cipher(algorithms.AES(key), modes.CBC(iv), default_backend())**: Εδώ, δημιουργείται ένα αντικείμενο Cipher χρησιμοποιώντας τον αλγόριθμο AES, το παρεχόμενο κλειδί και τον τρόπο λειτουργίας CBC (Cipher Block Chaining). Το algorithms.AES(key) καθορίζει τον αλγόριθμο AES με το δοσμένο κλειδί. Το modes.CBC(iv) καθορίζει τη λειτουργία CBC με το δεδομένο IV. Η default_backend() καθορίζει το προεπιλεγμένο κρυπτογραφικό backend.

- **encryptor = cipher.encryptor()**: Δημιουργείται ένα αντικείμενο encryptor από το αντικείμενο Cipher. Αυτό το αντικείμενο θα χρησιμοποιηθεί για την εκτέλεση της λειτουργίας κρυπτογράφησης.

- **padder =padding.PKCS7 (algorithmsȦESḃlock_size).padder()**: Αυτή η γραμμή δημιουργεί ένα αντικείμενο padder χρησιμοποιώντας το σχήμα padding PKCS7. Το padding.PKCS7 (algorithms.AES.block_size) καθορίζει το σχήμα padding PKCS7 με το μέγεθος μπλοκ του αλγορίθμου AES. Το αντικείμενο padder θα χρησιμοποιηθεί για να προσθέσει padding στα δεδομένα του απλού κειμένου πριν από την κρυπτογράφηση.

2 Κώδικας

Listing 1: Κώδικας κρυπτογράφησης AES με και χωρίς χρήση padding

```
import os,argparse
from cryptography.hazmat.primitives.ciphers import Cipher,algorithms,modes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.backends import default_backend

class Encryptor:
    def __init__(self,parser):
        args=parser.parse_args()
        folder=args.folder
        self.input_file=os.path.join(folder,args.input) if folder else args.input
        self.key_file=os.path.join(folder,args.key) if folder else args.key
        self.output_file=os.path.join(folder,args.output) if folder else args.outpute

    def encrypt_file_with_padding(self):
        key=None
        with open(self.key_file,'rb') as f:
            key=f.read()

        if key==None:
            raise ValueError("Key can not be setted to None value")

        #setup cipher: AES in CBC mode, with a random IV and no padding
        iv=os.urandom(algorithms.AES.block_size//8)
        cipher = Cipher(algorithms.AES(key), modes.CBC(iv), default_backend())
        encryptor = cipher.encryptor()
        padder=padding.PKCS7(algorithms.AES.block_size).padder()

        # Open input file for reading and output file for writing
        with open(self.input_file, 'rb') as f_in, open(self.output_file+"_with_padding", 'wb') as f_out:
            # Write the contents of the IV in the output file
            f_out.write(iv)

            while True:
                # Read a chunk of the input file into the plaintext variable
                plaintext = f_in.read(16)

                if not plaintext:
                    ciphertext = encryptor.update(padder.finalize())
```

```
                # Write the final ciphertext in the output file
                print(f'{ciphertext=}')
                f_out.write(ciphertext)
                break
            else:
                ciphertext = encryptor.update(plaintext)
                # Write the ciphertext in the output file
                f_out.write(ciphertext)

    def encrypt_file_no_padding(self):
        key=None
        with open(self.key_file,'rb') as f:
            key=f.read()

        if key==None:
            raise ValueError("Key can not be setted to None value")

        #setup cipher: AES in CBC mode, with a random IV and no padding
        iv=os.urandom(algorithms.AES.block_size//8)
        cipher = Cipher(algorithms.AES(key), modes.CBC(iv), default_backend())
        encryptor = cipher.encryptor()

        # Open input file for reading and output file for writing
        with open(self.input_file, 'rb') as f_in, open(self.output_file+"_without_padding", 'wb') as
             f_out:
            ciphertext=b""
            # Write the contents of the IV in the output file
            f_out.write(iv)

            while True:
                # Read a chunk of the input file into the plaintext variable
                plaintext = f_in.read(16)

                if not plaintext:
                    break

                plaintext_len=len(plaintext)

                # Truncate the plaintext to the nearest smaller multiple of the block size
                if plaintext_len % algorithms.AES.block_size != 0:
                    continue
                # plaintext = plaintext[:-(plaintext_len % algorithms.AES.block_size)]
                # print(len(plaintext))
                # if len(plaintext)%algorithms.AES.block_size!=0:
                # raise ValueError("Plaintext length must be a multiple of the block size")
                ciphertext+=encryptor.update(plaintext)
                f_out.write(ciphertext)
            ciphertext+=encryptor.finalize()
            f_out.write(ciphertext)
            print(f'{ciphertext=}')

    def encrypt_file(self,padding=False):
        if padding:
            self.encrypt_file_with_padding()
        else:
            self.encrypt_file_no_padding()

if __name__=='__main__':
    parser=argparse.ArgumentParser(description="Encrypt a file using AES encryption")
    parser.add_argument("--folder",required=True,help="Folder if data are in folder")
    parser.add_argument("--key",required=True,help="path to the key file")
    parser.add_argument("--input",required=True,help="path to the input file")
    parser.add_argument("--output",required=True,help="path to the output file")
    parser.add_argument("--with_padding",help="Using padding in the encryption",action='store_true')

    encryptor=Encryptor(parser)
    encryptor.encrypt_file(padding=True)
    encryptor.encrypt_file(padding=False)
```

Για εκτέλεση του κώδικα τρέχουμε την παρακάτω εντολή σε ένα τερματικό
python AES_encryption.py –folder data –key key_save –input passcode.txt –output

3 Ερωτήματα Εργασίας

Το μέγεθος των κρυπτογραφημένων αρχείων που προκύπτουν είναι μεγαλύτερο από το μέγεθος των αρχείων εισόδου λόγω της φύσης των αλγορίθμων κρυπτογράφησης, και συγκεκριμένα του αλγορίθμου κρυπτογράφησης AES σε αυτή την περίπτωση. Υπάρχουν μερικοί λόγοι για τους οποίους τα κρυπτογραφημένα αρχεία είναι πάντα μεγαλύτερα:

- **Μέγεθος μπλοκ:** Ο AES λειτουργεί σε μπλοκ δεδομένων σταθερού μεγέθους, συνήθως 128 bit (16 bytes) για τον AES-128. Εάν το μέγεθος του αρχείου εισόδου δεν είναι ακριβές πολλαπλάσιο του μεγέθους μπλοκ, προστίθεται συμπλήρωση για να γίνει πολλαπλάσιο. Στην περίπτωση του padding, εισάγονται επιπλέον bytes για να γεμίσουν το ελλιπές μπλοκ, αυξάνοντας το συνολικό μέγεθος του κρυπτογραφημένου αρχείου.

- **Διάνυσμα αρχικοποίησης (IV):** Το κρυπτογραφημένο αρχείο περιλαμβάνει το διάνυσμα αρχικοποίησης που χρησιμοποιείται για τη διαδικασία κρυπτογράφησης. Το IV είναι μια τυχαία τιμή που δημιουργείται για κάθε λειτουργία κρυπτογράφησης και απαιτείται για την αποκρυπτογράφηση. Το IV έχει συνήθως το ίδιο μέγεθος με ένα μπλοκ (16 bytes για τον AES-128). Η συμπερίληψη του IV στο κρυπτογραφημένο αρχείο αυξάνει το συνολικό μέγεθος του αρχείου.

- **Επιβάρυνση κρυπτογράφησης:** Το κρυπτογραφημένο αρχείο περιέχει επίσης πρόσθετα δεδομένα και μεταδεδομένα που σχετίζονται με τη διαδικασία κρυπτογράφησης, όπως αναγνωριστικά αλγορίθμων κρυπτογράφησης, πληροφορίες συμπλήρωσης (εάν υπάρχει) και τυχόν απαραίτητους ελέγχους ακεραιότητας ή ετικέτες ελέγχου ταυτότητας. Αυτά τα πρόσθετα bytes συμβάλλουν στο αυξημένο μέγεθος του κρυπτογραφημένου αρχείου.

- **Διαδικασία κρυπτογράφησης:** Η ίδια η διαδικασία κρυπτογράφησης εισάγει κάποιο επίπεδο επιβάρυνσης λόγω των μαθηματικών πράξεων που εμπλέκονται. Οι αλγόριθμοι κρυπτογράφησης, συμπεριλαμβανομένου του AES, χρησιμοποιούν πολύπλοκες πράξεις όπως αντικατάσταση, μετατροπή και πράξεις κατά δυφία που μπορεί να οδηγήσουν σε επέκταση των δεδομένων.

Συνολικά, αυτοί οι παράγοντες συμβάλλουν στην αύξηση του μεγέθους των κρυπτογραφημένων αρχείων σε σύγκριση με τα αρχικά αρχεία εισόδου. Η έκταση της αύξησης του μεγέθους εξαρτάται από παράγοντες όπως το μέγεθος του αρχείου εισόδου, το μέγεθος του μπλοκ που χρησιμοποιείται από τον αλγόριθμο κρυπτογράφησης και τυχόν συμπλήρωση ή πρόσθετα μεταδεδομένα που προστίθενται κατά τη διαδικασία κρυπτογράφησης.

Γενικά, ο αλγόριθμος κρυπτογράφησης AES λειτουργεί σε μπλοκ δεδομένων σταθερού μεγέθους, συνήθως 128 bit (16 bytes) για τον AES-128. Εάν το μέγεθος του αρχείου εισόδου δεν είναι πολλαπλάσιο του μεγέθους μπλοκ, απαιτείται κάποια μορφή συμπλήρωσης για να εξασφαλιστεί ότι τα δεδομένα μπορούν να χωριστούν σε πλήρη μπλοκ για κρυπτογράφηση.

Κατά την κρυπτογράφηση ενός αρχείου, το padding χρησιμοποιείται συνήθως για να επεκτείνει το μέγεθος του αρχείου στο πλησιέστερο πολλαπλάσιο του μεγέθους μπλοκ. Το padding προσθέτει επιπλέον bytes στα δεδομένα εισόδου για να γεμίσει τον εναπομείναντα χώρο στο τελευταίο μπλοκ. Αυτό διασφαλίζει ότι όλα τα μπλοκ μπορούν να υποβληθούν σε επεξεργασία από τον αλγόριθμο κρυπτογράφησης.

Χωρίς συμπλήρωση, εάν το μέγεθος του αρχείου εισόδου δεν είναι πολλαπλάσιο του μεγέθους μπλοκ, ο αλγόριθμος θα επεξεργαστεί μόνο τα πλήρη μπλοκ δεδομένων και θα απορρίψει τα υπόλοιπα bytes. Αυτό σημαίνει ότι το κρυπτογραφημένο κείμενο εξόδου θα έχει μήκος πολλαπλάσιο του μεγέθους του μπλοκ και το αρχικό μέγεθος του αρχείου δεν θα διατηρηθεί.

Είναι σημαντικό να λάβετε υπόψη σας τους περιορισμούς της μεθόδου κρυπτογράφησης και τις απαιτήσεις της συγκεκριμένης εφαρμογής σας. Εάν η διατήρηση του αρχικού μεγέθους του αρχείου αποτελεί απαίτηση, θα πρέπει να εξεταστούν εναλλακτικοί τρόποι κρυπτογράφησης ή τεχνικές που μπορούν να χειριστούν δεδομένα εισόδου αυθαίρετου μεγέθους χωρίς να απαιτείται συμπλήρωση. Αυτές οι εναλλακτικές προσεγγίσεις μπορούν να παρέχουν κρυπτογράφηση χωρίς να μεταβάλλουν το μέγεθος του αρχείου, διασφαλίζοντας ότι τα αποκρυπτογραφημένα δεδομένα αντιστοιχούν στην αρχική είσοδο.