ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
& ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

3η Εργασία

1.0 Αποκρυπτογράφηση ενός αρχείου με χρήση AES

Αναπτύξτε ένα πρόγραμμα παρόμοιο με το προηγούμενο αλλά ικανό να αποκρυπτογραφήσει ένα αρχείο. Θα πρέπει να αποσταλούν τρία ορίσματα στο πρόγραμμα: το αρχείο κλειδί(key file), το αρχείο προς αποκρυπτογράφηση(file to decipher) και το αρχείο εξόδου(output file).

Ακολουθεί ο κώδικας σε Java

```
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.SecretKey;
import java.security.SecureRandom;
import javax.crypto.Cipher;

// Read key bytes from key file (into variable keyBytes)
...

// Setup AES key with the key to encrypt

SecretKey key = new SecretKeySpec( keyBytes, "AES" );

// Create a cipher engine given the algorithm (AES), the encryption mode (CBC)
// and the padding (PKCS \#5)

Cipher c = Cipher.getInstance( "AES/CBC/PKCS5Padding" );

// Set the IV parameters (required for CBC) to a random value.
// The IV must have a size equal to the cipher's block size

byte[] iv = new byte[c.getBlockSize()];
SecureRandom random = new SecureRandom();
random.nextBytes( iv );

// Set the cipher engine to encrypt with the intended key and IV

c.init( Cipher.ENCRYPT_MODE, key, new IvParameterSpec( iv ) );

// Open input file for reading and output file for writing
...

// Write the IV in the output file
...

while (...) { // Cicle to repeat while there is data left on the input file
    // Read a chunk of the input file to the plaintext variable
    ...

    // The length of the plaintext data should be stored in pLen

    ciphertext = c.update( plaintext, 0, pLen );

    // Store the ciphertext in the output file
    ...
}

// Perform the encryption of the last plaintext contents + the padding

ciphertext = c.doFinal();

// Store the ciphertext in the output file
```

Ακολουθεί ο κώδικας σε Python

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

# Read key bytes from key file (into variable key)
...

# Setup cipher: AES in CBC mode, w/ a random IV and PKCS #7 padding (similar to PKCS #5

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
...

# Write the contents of iv in the output file
...

while True: # Cicle to repeat while there is data left on the input file

    # Read a chunk of the input file to the plaintext variable
    ...

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

        # Write the contents of ciphertext in the output file

        break
    else:
        ciphertext = encryptor.update( padder.update( plaintext ) )

        # Write the ciphertext in the output file
```

Ακολουθεί ο κώδικας σε C

```c
#include <stdint.h>
#include <openssl/crypto.h>
#include <openssl/evp.h>
#include <openssl/aes.h>
#include <openssl/rand.h>

#define KEY_LEN 16 /* bytes */

EVP_CIPHER_CTX * ctx;
uint8_t key[KEY_LEN];
uint8_t iv[AES_BLOCK_SIZE];

// Read key bytes from key file (into variable key)
...

// Setup cipher: AES in CBC mode, w/ a random IV and PKCS #7 padding (similar to PKCS #5)

RAND_bytes( iv, sizeof(iv) );
ctx = EVP_CIPHER_CTX_new();
EVP_CipherInit( ctx, EVP_aes_128_cbc(), key, iv, 1 );

// PKCS #7 padding is on by default; use the following call to disable it
// EVP_set_padding( ctx, 0 );

// Open input file for reading and output file for writing
...

// Write the contents of iv in the output file
...

while (...) {
    uint8_t ciphertext[ /* allocate as much bytes as for variable plaintext (>=
        AES_BLOCK_SIZE) */ ];
    int cLen;

    // Read a chunk of the input file to the plaintext variable
    ...

    // The length of the plaintext data should be stored in pLen

    EVP_CipherUpdate( ctx, ciphertext, &cLen, plaintext, pLen );

    // Write the first cLen bytes of ciphertext in the output file
    ...
}

EVP_CipherFinal( ctx, ciphertext, &cLen );
// Write the first cLen bytes of ciphertext in the output file
...

EVP_CIPHER_CTX_free( ctx );
```

Συμβουλή: Προσπαθήστε να χρησιμοποιήσετε ένα πρόγραμμα γραμμένο σε μια διαφορετική γλώσσα από αυτή με την οποία κάνατε το κρυπτογράφηση

2.0 Κρυπτογράφηση και αποκρυπτογράφηση ενός αρχείου χρησιμοποιώντας έναν δεδομένο αλγόριθμο

Τροποποιήστε τα προηγούμενα προγράμματα ώστε να δέχονται μια πρόσθετη παράμετρο που δηλώνει τον αλγόριθμο που θα χρησιμοποιηθεί.

Σκεφτείτε ότι μπορούν να δοθούν δύο επιλογές: **AES και DES**. Η αλλαγή θα πρέπει να είναι ελάχιστη.

3.0 Λειτουργίες κρυπτογράφησης

Ορισμένοι τρόποι κρυπτογράφησης που απαιτούν πληροφορίες ανατροφοδότησης (CBC, OFB, CFB και CTR) πρέπει να χρησιμοποιούν ένα διάνυσμα αρχικοποίησης (IV).

Τροποποιήστε τα προηγούμενα προγράμματα ώστε να δέχονται ως όρισμα ένα όνομα αρχείου, το οποίο θα πρέπει να περιέχει έναν ορισμό του αλγορίθμου κρυπτογράφησης, τον τρόπο κρυπτογράφησης και ένα IV (υποθέστε ένα προεπιλεγμένο γέμισμα όταν απαιτείται). Το περιεχόμενο αυτού του αρχείου θα πρέπει να δημιουργείται από την εφαρμογή κρυπτογράφησης και να χρησιμοποιείται από την εφαρμογή αποκρυπτογράφησης για την αρχικοποίηση της μηχανής αποκρυπτογράφησης. Το IV σε αυτό το αρχείο θα πρέπει να είναι μια τυχαία τιμή που παράγεται κατά την κρυπτογράφηση.

Σημείωση: δεν χρειάζεται πλέον να αποθηκεύετε το IV στην αρχή του κρυπτογραφημένου αρχείου. Λάβετε υπόψη ότι μόνο η λειτουργία κρυπτογράφησης ECB δεν απαιτεί τη χρήση IV.

4.0 Διάδοση μοτίβων(Propagation of patterns)

Σε αυτή την άσκηση ο στόχος είναι να αναλυθεί ο αντίκτυπος της χρήσης ECB και CBC από την άποψη της διάδοσης των μοτίβων μεταξύ του απλού κειμένου και του αντίστοιχου κρυπτογραφήματος.

Η προσέγγιση που θα ακολουθηθεί θα είναι η χρήση ενός αρχείου BMP, η κρυπτογράφησή του και η οπτικοποίηση του κρυπτογραφήματος που προκύπτει. Η μορφή BMP είναι πολύ απλή και εφόσον τα πρώτα 54 bytes (η επικεφαλίδα) διατηρούνται αμετάβλητα, το υπόλοιπο περιεχόμενο θα εμφανίζεται ως εικόνα.

Με το πρόγραμμα που αναπτύξατε, και χρησιμοποιώντας τη λειτουργία ECB, με οποιονδήποτε αλγόριθμο κρυπτογράφησης, κρυπτογραφήστε το αρχείο security.bmp σε ένα άλλο αρχείο, με όνομα security-ecb.bmp. Στη συνέχεια, επαναφέρετε την επικεφαλίδα BMP του κρυπτογραφημένου αρχείου με την αρχική. Αυτό θα επιτρέψει σε οποιαδήποτε εφαρμογή γραφικών να ερμηνεύσει το αρχείο ως αρχείο BMP, ακόμη και αν τα δεδομένα της εικόνας είναι κρυπτογραφημένα.

Η ακόλουθη εντολή μπορεί να χρησιμοποιηθεί για να διορθώσετε την επικεφαλίδα από τα κρυπτογραφημένα περιεχόμενα στα αρχικά:

```
dd if=security.bmp of=security-ecb.bmp bs=1 count=54 conv=notrunc
```

Ανοίξτε και τους δύο φακέλους με οποιοδήποτε πρόγραμμα προβολής BMP και συγκρίνετε τα αποτελέσματα.
Επαναλάβετε την ίδια διαδικασία με τον ίδιο αλγόριθμο και πάνω από το ίδιο fle security.bmp, αλλά τώρα χρησιμοποιώντας τη λειτουργία CBC. Θα πρέπει να δημιουργήσετε ένα αρχείο με όνομα security-cbc.bmp. Επαναφέρετε την επικεφαλίδα, δείτε και τις δύο εικόνες και συγκρίνετε το αποτέλεσμα. Επαναλάβετε τα παραπάνω βήματα για άλλους αλγορίθμους και τρόπους κρυπτογράφησης. Τι μπορείτε να συμπεράνετε;

5.0 Διάδοση σφαλμάτων(Error propagation)

Σε αυτή την άσκηση θα αναλύσουμε τον αντίκτυπο των σφαλμάτων στο κρυπτογραφημένο κείμενο. Δηλαδή, την επίδραση των τροποποιήσεων σε ένα ή περισσότερα bits του κρυπτοκειμένου, και στη συνέχεια την αποκρυπτογράφηση του κρυπτοκειμένου στο καθαρό κείμενο, όταν χρησιμοποιείται ECB, CBC, OFB και CFB.

Χρησιμοποιώντας το πρόγραμμα που αναπτύχθηκε, με οποιονδήποτε αλγόριθμο κρυπτογράφησης, και τον τρόπο κρυπτογράφησης ECB, κρυπτογραφήστε την εικόνα σας. ***Μην επαναφέρετε την επικεφαλίδα!***

Χρησιμοποιώντας έναν επεξεργαστή δεκαεξαδικών, όπως το ghex, επιλέξτε ένα τυχαίο byte και λάβετε υπόψη σας αυτό το byte. Στη συνέχεια, αναστρέψτε ένα bit στην αντίθετη τιμή. Ως παράδειγμα, μπορείτε να χρησιμοποιήσετε τη διεύθυνση 0xec00 η οποία κωδικοποιεί το κάτω δεξιά

εικονοστοιχείο της τελείας στο θαυμαστικό, μετά τη λέξη RSA.

Αποκρυπτογραφήστε το αρχείο που μόλις τροποποιήσατε χρησιμοποιώντας τον ίδιο αλγόριθμο και τρόπο. Δείτε τόσο την αρχική εικόνα, όσο και αυτή που μόλις πήρατε. Στη συνέχεια, συγκρίνετε το περιεχόμενό τους χρησιμοποιώντας έναν επεξεργαστή δεκαεξαδικών. Συγκεκριμένα, εστιάστε στο στο byte που μόλις αλλάξατε και στα γύρω bytes. Μπορείτε επίσης να χρησιμοποιήσετε την εντολή

cmp -bl first File secondFile

Επαναλάβετε αυτά τα βήματα με τους υπόλοιπους τρόπους κρυπτογράφησης, και για κάθε έναν και ποια είναι η επίπτωση των σφαλμάτων στο κρυπτογράφημα. Επίσης, προσδιορίστε ποιοι τρόποι κρυπτογράφησης είναι περισσότερο και ποιοι λιγότερο ευαίσθητοι σε σφάλματα στο κρυπτοκείμενο. (λαμβάνοντας υπόψη την ποσότητα των σφαλμάτων στην τελική εικόνα).

6.0 Triple DES

Ο αλγόριθμος DES χρησιμοποιεί κλειδιά με 56 bit και θεωρήθηκε ανασφαλής λίγο καιρό μετά τη δημιουργία του. Ωστόσο, δημιουργήθηκε μια μέθοδος για την αύξηση της ασφάλειάς του διπλασιάζοντας ή τριπλασιάζοντας το μέγεθος του κλειδιού. Αυτό το μέθοδος ονομάζεται συχνά TripleDES ή 3DESede. Αυτό που εισάγει αυτή η μέθοδος είναι η έννοια του πολλαπλών λειτουργιών πάνω στο κείμενο, με τη χρήση διαφορετικών κλειδιών και αποτελεί ένα καλό παράδειγμα ενίσχυσης της κρυπτογράφησης.

Όταν χρησιμοποιούνται δύο κλειδιά (112 bits), το TripleDES υλοποιείται με τον υπολογισμό:

Ek1 (Dk2 (Ek1 (text)))

Όταν χρησιμοποιούνται τρία κλειδιά (168 bit), το TripleDES υλοποιείται με τον υπολογισμό:

Ek3 (Dk2 (Ek1 (text)))

Όπου k_i είναι ένα κλειδί, D είναι μια λειτουργία αποκρυπτογράφησης και E είναι μια λειτουργία κρυπτογράφησης. Αυτή η μέθοδος βασίζεται σε στο γεγονός ότι η αποκρυπτογράφηση ενός κρυπτογραφήματος με λάθος κλειδί ισοδυναμεί με την εκ νέου κρυπτογράφησή του.

Υλοποιήστε ένα πρόγραμμα που εφαρμόζει αυτή τη μέθοδο στην κρυπτογράφηση DES(java,python,C). Λάβετε υπόψη ότι, κατά την κρυπτογράφηση, μόνο η πρώτη λειτουργία κρυπτογράφησης πρέπει να χρησιμοποιεί γέμισμα! Κατά την αποκρυπτογράφηση, η τελευταία πράξη θα πρέπει να χρησιμοποιεί το padding[1].

1 Η Java, η Python και η C υποστηρίζουν εγγενώς το 3DESede, αλλά δεν θα πρέπει να το χρησιμοποιήσετε, εκτός αν θέλετε να δοκιμάσετε την υλοποίησή σας.

7.0 Απόδοση κρυπτογράφησης(Cipher performance)

Μια σημαντική πτυχή των διαφορετικών κρυπτογραφήσεων είναι η απόδοσή τους σε κοινό υλικό, η οποία ποικίλλει σε μεγάλο βαθμό. Λαμβάνοντας υπόψη τα κρυπτογραφήματα που είναι διαθέσιμα στη Java (Blowfish, AES, DES, RC2, RC4, ARCFOUR και 3DESede), υλοποιούμε ένα πρόγραμμα για τη συγκριτική αξιολόγηση κάθε κρυπτογράφησης. Θεωρήστε μπλοκ με μέγεθος που κυμαίνεται από 16 bytes έως 8192 bytes. Για να εκτελέσετε το benchmark, θεωρήστε τη μέθοδο System.currentTimeMillis() και δείτε πόσες ώρες χρειάζονται για να γίνουν 10, 000, 000 λειτουργίες κρυπτογράφησης.

Συμβουλή: μην λάβετε υπόψη για τον χρονοπρογραμματισμό κατα την πρώτη και δεύτερη λειτουργία κρυπτογράφησης/αποκρυπτογράφησης, γιατί θα βγάλει λάθος αποτελέσματα