






Εργασία στην Συμμετρική Κρυπτογραφία

Στην εργασία αυτή απαιτείται η εγκατάσταση του περιβάλλοντος ανάπτυξης Java (JDK), η Python 3 ή ο μεταγλωττιστής και το περιβάλλον ανάπτυξης της C.



Τα παραδείγματα που παρέχονται θα χρησιμοποιήσουν τόσο τη Java, όσο και την Python και τη C. Για την Python θα πρέπει να εγκαταστήσετε το cryptography module. Για τη C θα χρησιμοποιήσουμε τις συναρτήσεις EVP (Digital Envelope) από το Crypto βιβλιοθήκη που αποτελεί μέρος του OpenSSL, και γι' αυτό θα πρέπει να χρησιμοποιήσετε την εγκατάσταση των πακέτων libssl1.0-dev και libssl1.1.

Ενδιαφέροντα στοιχεία

Java

-  **javax.crypto.Cipher:** Μια περίπτωση του Cipher επιτρέπει την κρυπτογράφηση και την αποκρυπτογράφηση. Ορισμένες σημαντικές μέθοδοι είναι: getInstance, init, update και doFinal.
-  **javax.crypto.KeyGenerator:** Μια περίπτωση της κλάσης KeyGenerator είναι μια γεννήτρια συμμετρικών κλειδιών. Ορισμένες σημαντικές μέθοδοι είναι: getInstance, init και generateKey.
-  **javax.crypto.SecretKey:** Ένα παράδειγμα μιας κλάσης που υλοποιεί τη διεπαφή SecretKey είναι ένα συμμετρικό κλειδί

Python

-  Το module κρυπτογραφίας- cryptography module είναι ταυτόχρονα ένα front-end, μια διεπαφή υψηλού επιπέδου για την αντιμετώπιση της κρυπτογραφίας όσο και μια προεπιλεγμένη υλοποίηση των θεμελιωδών κρυπτογραφικών πρωτοβουλιών. Η διεπαφή υψηλού επιπέδου επιτρέπει τη διερεύνηση ενός backend διαφορετικού από το προεπιλεγμένο. Μια αναφορά στο προεπιλεγμένο backend λαμβάνεται με τη συνάρτηση default_backend.
-  Ο διερμηνέας της Python διακρίνει τα bytes από τους χαρακτήρες κειμένου, και συνεπώς τους πίνακες byte από τους συμβολοσειρές. Μια συμβολοσειρά είναι κάτι περισσότερο από ένας πίνακας byte, έχει επίσης μια κωδικοποίηση (π.χ. UTF-8, Unicode κ.λπ.). Μπορείτε να πάρετε τα bytes των χαρακτήρων (συμβολοσειρών) χρησιμοποιώντας τη



συνάρτηση bytes και μπορείτε να μετατρέψετε τα bytes σε χαρακτήρες (συμβολοσειρές) χρησιμοποιώντας τη μέθοδο `decode` σε έναν πίνακα byte (με μια κωδικοποίηση στόχου).

Σημείωση: Από προεπιλογή, στην κρυπτογραφία χρησιμοποιούμε πάντα bits και bytes, τίποτα άλλο. Επομένως, μην επιχειρήσετε να λύσετε τα σφάλματα του διερμηνέα αλλάζοντας τα πάντα σε κείμενο (χαρακτήρες και συμβολοσειρές)!

C

- ✚ Οι συναρτήσεις **EVP_- - -_ex** είναι συναρτήσεις που μπορούν να παραμετροποιηθούν με μια μηχανή υλοποίησης, ενώ οι συναρτήσεις χωρίς το **_ex** χρησιμοποιούν την προεπιλεγμένη μηχανή της βιβλιοθήκης.
- ✚ Οι λειτουργίες **EVP_CipherInit**, **EVP_CipherUpdate** και **EVP_CipherFinal** αποτελούν τη διεπαφή υψηλού επιπέδου για το χειρισμό των λειτουργιών κρυπτογράφησης και αποκρυπτογράφησης.
- ✚ Οι συναρτήσεις **EVP_Encrypt- -** αποτελούν τη διεπαφή υψηλού επιπέδου για το χειρισμό κρυπτογραφήσεων, ενώ οι συναρτήσεις **EVP_- Decrypt- -** χειρίζονται τις αποκρυπτογραφήσεις (χρησιμοποιώντας τις προηγούμενες).
- ✚ Οι συναρτήσεις **EVP_CIPHER_CTX_set- -** βοηθούν στην παραμετροποίηση ενός πλαισίου κρυπτογράφησης, ενδεχομένως με στοιχεία που δεν είναι διαθέσιμα στις συναρτήσεις **EVP_- -Init**.
- ✚ Μια δομή **EVP_CIPHER_CTX** περιγράφει το τρέχον πλαίσιο ενός κρυπτογραφικού μετασχηματισμού και δημιουργείται με την **EVP_CIPHER_CTX_new** και αποδεσμεύεται με την **EVP_CIPHER_CTX_free**. Αυτό το πλαίσιο είναι χρησιμοποιείται για την αποθήκευση θεμελιωδών στοιχείων δεδομένων (π.χ. κλειδί και IV) που χρησιμοποιούνται για την επεξεργασία κρυπτογραφικών μετασχηματισμών. και για την αποθήκευση δεδομένων εισόδου για την αντιμετώπιση της ευθυγράμμισης μπλοκ και της συμπλήρωσης.
- ✚ Μια δομή **EVP_CIPHER** περιγράφει γενικά χαρακτηριστικά ενός συγκεκριμένου κρυπτογραφικού μετασχηματισμού. Συνήθως δεν χρειάζεται να συμπληρώνεται χειροκίνητα, μπορεί κανείς να χρησιμοποιήσει προκαθορισμένες συναρτήσεις γι' αυτό (π.χ. Η **EVP_aes_256_cbc** επιστρέφει έναν δείκτη σε μια τέτοια δομή που περιγράφει έναν κρυπτογράφο AES με 256-bit. κλειδί που λειτουργεί σε λειτουργία CBC).



Δημιουργία συμμετρικού κλειδιού από έναν κωδικό πρόσβασης

Αναπτύξτε ένα πρόγραμμα για τη δημιουργία ενός συμμετρικού κλειδιού για τον αλγόριθμο AES και αποθηκεύστε το σε ένα αρχείο. Το κλειδί θα πρέπει να παράγεται από έναν κωδικό πρόσβασης, μια διαδικασία γνωστή ως κρυπτογράφηση με βάση τον κωδικό πρόσβασης (Password-Based Encryption - PBE). Ο κωδικός πρόσβασης θα πρέπει να παρέχεται ως πρώτο όρισμα του προγράμματος και το όνομα του αρχείου ως δεύτερο.

Συμβουλή: μπορείτε να χρησιμοποιήσετε λιγότερα ορίσματα και να εξετάσετε τη χρήση του `stdout` σε περίπτωση απουσίας προδιαγραφών του αρχείου.

Το πρόγραμμα θα πρέπει να παράγει κλειδιά 128-bit, καθώς αυτό απαιτείται από τον αλγόριθμο AES. Για την αντιμετώπιση του PBE θα χρησιμοποιήσουμε μια συνάρτηση μετατροπής κωδικού πρόσβασης σε κλειδί, γνωστή ως PBKDF2 (Password-Based Key Derivation Function 2). Πρόκειται για μια γενική συνάρτηση κατακερματισμού πολλαπλών επαναλήψεων, η οποία μπορεί να παραμετροποιηθεί με μια άλλη συνάρτηση κατακερματισμού με κλειδί- εμείς θα χρησιμοποιήσουμε την HMAC με SHA-1.

Συμβουλή: μπορείτε να χρησιμοποιήσετε λιγότερα ορίσματα και να εξετάσετε τη χρήση του `STDOUT` ελλείψει προδιαγραφών αρχείου.

Συμβουλή: η έξοδος θα πρέπει να είναι ανεξάρτητη από τη γλώσσα, επομένως θα πρέπει να έχετε ακριβώς την ίδια έξοδο χρησιμοποιώντας διαφορετικές γλώσσες προγραμματισμού.

Σημείωση: συναρτήσεις όπως η PBKDF2 χρησιμοποιούνται συχνά για τον υπολογισμό μετασχηματισμών κωδικών πρόσβασης που χρησιμοποιούνται σε διαδικασίες ελέγχου ταυτότητας. Ο τελικός στόχος είναι να αποφεύγεται η αποθήκευση κωδικών πρόσβασης σε καθαρό κείμενο για την αντιστοίχιση με αυτούς που παρέχουν οι άνθρωποι. Ωστόσο, για την αποτροπή επιθέσεων εξαντλητικής αναζήτησης που αναζητούν έναν κωδικό πρόσβασης κατάλληλο για έναν δεδομένο αποθηκευμένο μετασχηματισμό, οι εν λόγω μετασχηματισμοί συχνά τυχαioποιούνται με ένα στοιχείο που ονομάζεται αλάτι.

Ένα salt δεν χρειάζεται να είναι μυστικό, αρκεί να εκχωρείται τυχαία μία φορά για έναν δεδομένο κωδικό πρόσβασης και να χρησιμοποιείται για τον μετασχηματισμό του στη συνέχεια- με αυτόν τον τρόπο, μπορούμε να σπάσουμε μαζικές επιθέσεις μαντεψιάς κωδικών πρόσβασης, όπως αυτή που μπορεί κανείς να υλοποιήσει με πίνακες ουράνιου τόξου, επειδή είναι ανέφικτο για αυτές να

να αντιμετωπίσουν διαφορετικές τιμές αλατιού. Τα ακόλουθα δείγματα κώδικα παρουσιάζουν ενδεικτικά πώς να μετατρέψετε έναν κωδικό πρόσβασης κειμένου, αποθηκευμένο στη



συμβολοσειρά pwd, σε πίνακα byte (κλειδί) με PBKDF2 (με HMAC και SHA-1) χρησιμοποιώντας Java, Python και C.

Ακολουθεί ο κώδικας σε Java

```
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.SecretKeyFactory;
import javax.crypto.SecretKey;

// PBKDF2 specifications: password, salt, iterations and output size
//
// salt: a byte array intended to produce different outputs for the same password
// iterations: the number of times the PBKDF2 iterates internally over a generator function
// output size: the number of bits we want to generate from the password (128 for AES)

byte[] salt = new byte[1];
salt[0] = 0; // We need to provide one salt ...

// First, we create a specifications object with all the PBKDF2 parameters we want
PBEKeySpec pbeKeySpec = new PBEKeySpec( pwd.toCharArray(), salt, 1000, 128);

// Then, we create key material (a byte array) using PBKDF2 + HMAC(SHA-1)
SecretKeyFactory skf = SecretKeyFactory.getInstance( "PBKDF2WithHmacSHA1" );
byte[] key = skf.generateSecret( pbeKeySpec ).getEncoded();

// Write key in a file
```

Ακολουθεί ο κώδικας σε Python

```
import sys
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.backends import default_backend

# The PBKDF2 generator of Python receives as input the number of bytes to generate,
# instead of bits

salt = b'\x00'
kdf = PBKDF2HMAC( hashes.SHA1(), 16, salt, 1000, default_backend() )
key = kdf.derive( bytes( pwd, 'UTF-8' ) )

# Write key in a file
```

Ακολουθεί ο κώδικας σε C

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

#define KEY_LEN 16 /* bytes */

char salt = 0;
uint8_t key[KEY_LEN];

// The PBKDF2 generator of OpenSSL receives as input the number of bytes to generate,
// instead of bits

PKCS5_PBKDF2_HMAC_SHA1( pwd, -1, &salt, 1, 1000, sizeof(key), key );

// Write key in a file
```
