ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ Ασφάλεια Πληροφοριακών Συστημάτων (**ΠΛΗ591**) – 2ο σετ προβλημάτων

Παρακαλώ προσέξτε ότι όλες οι παρακάτω ασκήσεις πρέπει να έχουν παραδοθεί μέχρι τις 29/11/2015 (23:50).

REV 0.2

Σημαντικές παρατηρήσεις:

1. Για τα παρακάτω προβλήματα προτείνεται να χρησιμοποιήσετε γλώσσα προγραμματισμού **Python 2.7** σε πλατφόρμα **Linux** (Οδηγίες στο τέλος του εγγράφου).
2. Η επίλυση απορίων σχετικά με τα παρακάτω προβλήματα γίνεται αποκλειστικά μέσω του forum στο courses (στα Threads «SET 2: ProblemXX»). Όχι με προσωπικά e-mails.
3. Οποιαδήποτε άλλη ερώτηση σχετικά με το μάθημα πρέπει να απευθύνεται στο e-mail του διδάσκοντα.
4. Θα πρέπει να αποστείλετε τον κώδικα σας σε **1 απλό source[1] για κάθε Πρόβλημα (problemXX.py**).
5. Χρησιμοποιείστε σχόλια και τα απαραίτητα «prints» όπου χρειάζεται έξοδος.
6. **Δεν απαιτείται ξεχωριστό report.**
7. Στο **ίδιο zip file** θα βρίσκονται και τα 4 προβλήματα **σαν ξεχωριστά sources** μαζί με **όλα τα υπόλοιπα αρχεία** που χρειάζεται ο κώδικας σας για να τρέξει. Π.χ plaintext file, RSA keys κτλ.
8. Ο κώδικας θα πρέπει να τρέχει **"out-of-the-box"!** χωρίς καμία τροποποίηση.
9. Στην αρχή του κώδικα θα αναφέρονται όλα τα προσωπικά σας στοιχειά σας και το email σας σαν σχόλια καθώς και τα dependencies του κώδικα σας (αν υπάρχουν).
10. Όλοι οι κώδικες σας θα ελεγθούν για ομοιότητες.

Πρόβλημα 1:

Το ακόλουθο πρόβλημα θα σας εξοικειώσει με τις διάφορες βιβλιοθήκες κρυπτογραφίας.

Δημιουργείστε ένα πρόγραμμα που να επιτρέπει στους χρήστες να κρυπτογραφούν, αποκρυπτογραφούν, να κάνουν hash και να υπογράφουν αρχεία και να επιβεβαιώνουν υπογραφές αρχείων. Η κρυπτογράφηση/αποκρυπτογράφηση θα χρησιμοποιεί AES σε CBC mode. Το συνάρτηση hash θα χρησιμοποιεί SHA-256. Για την υπογραφή θα χρησιμοποιεί SHA-256 για το hash-ing του

[1] Δεν προτείνεται να έχετε πολλαπλά include files. Χρησιμοποιώντας python ο κώδικάς σας θα είναι σύντομος και κατανοητός.

αρχείου και RSA για την υπογραφή του. Το πρόγραμμα θα απεικονίζει ένα απλό μενού στο οποίο θα παρουσιάζονται στο χρήστη αυτές οι τρεις επιλογές σαν λίστα. Για την κάθε επιλογή το input θα είναι ως εξής:

1. Κρυπτογράφηση: Όνομα αρχείου και 16 byte κλειδί
2. Αποκρυπτογράφηση: Όνομα αρχείου και 16 byte κλειδί
3. Hash: Όνομα αρχείου
4. Υπογραφή: Το όνομα του αρχείου που πρόκειται να υπογραφεί και το όνομα του αρχείου που περιλαμβάνει το RSA ιδιωτικό κλειδί.
5. Επιβεβαίωση υπογραφής: Το όνομα του αρχείου που πρόκειται να υπογραφεί , το όνομα του αρχείου που περιλαμβάνει την υπογραφή και το όνομα του αρχείου που περιλαμβάνει το RSA δημόσιο κλειδί.

Θα χρειαστεί να δημιουργήσετε RSA κλειδιά για να τα χρησιμοποιήσετε στο πρόγραμμα. Για τις πρώτες 4 επιλογές, το αποτέλεσμα να γραφεί σε ένα αρχείο. Για την 5η επιλογή, το πρόγραμμα στέλνει μια ένδειξη για το εάν ήταν ή δεν ήταν σωστή η υπογραφή.

Σημείωση: Για τη δική σας διευκόλυνση **μην χρησιμοποιήσετε ορίσματα από γραμμή εντολών**. Φτιάξτε hard-coded τα ονόματα των αρχείων που θα διαβάζετε/γράφετε μιας και το παραδοτέο σας θα περιλαμβάνει **όλα τα παρελκόμενα** αρχεία που απαιτεί.

Πρόβλημα 2:

Το δεύτερο πρόβλημα έχει ως στόχο να σας εξοικειώσει με τις βιβλιοθήκες TLS (ssl).

Δημιουργείστε έναν client και έναν server ο οποίος συνδέεται μέσω SSL. Μόνο ο server είναι πιστοποιημένος (one-way authentication). Αφού ο client συνδεθεί με το server, ο client θα προτρέψει το χρήστη για μια γραμμή εισόδου την οποία θα στείλει στον server. Ο server θα λάβει το κείμενο από τον client και θα το τυπώσει στην οθόνη. Μπορείτε να χρησιμοποιήσετε self-signed certificates ή να δημιουργήσετε μια CA για να υπογράψετε το πιστοποιητικό (δε χρειάζεται να πάρετε πιστοποίηση υπογεγραμμένη από πραγματικό CA).

Σημείωση: Επειδή ο server και ο client θα τρέχουν σε διαφορετικό τερματικό, θα χρησιμοποιήσετε διαφορετικά source files για το συγκεκριμένο πρόβλημα. Π.χ problem2_client.py, problem2_server.py

TIP: https://pymotw.com/2/socket/tcp.html

Πρόβλημα 3:

Υλοποιείστε τον AES αλγόριθμο κρυπτογράφησης και τις συναρτήσεις επέκτασης κλειδιού (χρησιμοποιώντας έτοιμο κώδικα που μπορείτε να βρείτε σε διάφορα έτοιμα πακέτα). Θα χρειαστεί να τον τροποποιήσετε για να καταγράφει το χρόνο και να τρέχει το απαραίτητο κομμάτι(α).

Τρέξτε τη συνάρτηση κρυπτογράφησης 1 εκατομμύριο φορές (κάντε hard-code ένα απλό κείμενο, και το κλειδί, τρέξτε το κλειδί επέκτασης και μετά κάντε loop στη της συνάρτησης κρυπτογράφησης 1 εκατομμύριο φορές).

- Ποιος είναι ο μέσος χρόνος που χρειάζεται για να κρυπτογραφήσει ένα απλό κείμενο;

Τρέξτε την επέκταση κλειδιού 1 εκατομμύριο φορές (μην τρέξετε τη συνάρτηση κρυπτογράφησης).

- Ποιος είναι ο μέσος χρόνος που χρειάζεται για να πραγματοποιηθεί η επέκταση ενός κλειδιού;

Σημείωση: Στο αρχείο σας problem3.py θα αναφέρεται η original πηγή του source που χρησιμοποιήσατε καθώς και ποιες αλλαγές κάνατε. Τους χρόνους που πήρατε στο μηχάνημα σας μπορείτε επίσης να τους συμπεριλάβετε με σχόλια στον κώδικα σας.

TIP1: https://github.com/bozhu/AES-Python/blob/master/aes.py

TIP2: https://docs.python.org/2/library/timeit.html

Πρόβλημα 4:

Απλά στατιστικά τεστ σε AES.

Το κλειδί θα διατηρηθεί σταθερό κατά τη διάρκεια αυτού του πειράματος. Ορίστε με hard-code ένα κλειδί 128-bit με όλα τα bits= ‘0’. Επεξεργαστείτε τον AES κώδικα ώστε να μπορείτε να ελέγχετε πόσους γύρους τρέχει (όλοι οι γύροι θα έχουν όλα τα βήματα). Θα τρέξετε τον AES από τον 1 μέχρι και τον 4 γύρο).

Για τον αριθμό των γύρων=1,2,3,4 κάντε τα ακόλουθα:

- Κρυπτογραφείστε 129 διαφορετικά 128-bit απλά κείμενα, τα οποία προκύπτουν με τον **ακόλουθο τρόπο**:

- Το πρώτο απλό κείμενο θα είναι όλα τα bits=’0’ και κάθε επόμενο απλό κείμενο θα έχει ακριβώς ένα ‘1’ μέσα του συμφώνα με τον ακόλουθο κανόνα και με αυστηρή σειρά:
- 1ο απλό κείμενο->all zeros
- Κάθε επόμενο απλό κείμενο θα ξεκινάει με ένα bit “1” ξεκινώντας από το λιγότερο σημαντικό bit και σε κάθε επανάληψη το bit= “1” ολισθαίνει κατά μια θέση μέχρι να παραχθούν 128 διαφορετικά απλά κείμενα.

Τα κρυπτογραφείτε όλα και στο τέλος της διαδικασίας θα έχετε 129 ciphertexts. Ενώστε τα με τη σειρά που δημιουργήθηκαν σαν ένα συνεχόμενο bitstream.

Για κάθε έναν από τους γύρους 1,2,3,4 υπολογίστε τα ακόλουθα στατιστικά πάνω στο ciphertext-bitstream (θα χρειαστεί να γράψετε κάποιο κώδικα γι ’αυτό):

1. Ποια είναι η συχνότητα εμφάνισης ‘1’ στο bitstream εξεφρασμένη σε ποσοστό επί των συνολικών bits.
2. Πόσα“runs of ones” με μήκος 2 (δηλαδή αλληλουχιών “110”) εμφανίζονται στο bitstream.
3. Πόσα“runs of zeros” με μήκος 2 (δηλαδή αλληλουχιών “001”) εμφανίζονται στο bitstream.

Οδηγίες και χρήσιμα Links για Python:

Χρησιμοποιώντας Python θα έχετε ταχύτερους χρόνους σε prototyping και debugging. Ταυτόχρονα θα έχετε πολύ καλύτερη αίσθηση των δεδομένων και των μετασχηματισμών που γίνονται σε αυτά, μιας και η Python είναι interpreter γλώσσα και σας επιτρέπει πολύ εύκολα να διακόψετε τη ροή του προγράμματος και να διαβάσετε τις μεταβλητές σας.

Συνοπτικά:

1. Χρησιμοποιήστε Virtual Environment (VE): ***Anaconda (Python 2.7)***
    a. https://www.continuum.io/downloads
    b. Χρησιμοποιήστε το Development Environment ***Spyder*** το οποίο περιλαμβάνεται στο ***Anaconda*** (απλώς γράψτε $spyder στη γραμμή εντολών)
    c. Μπορείτε να ανανεώσετε οποιαδήποτε πακέτα με απλές εντολές μέσω του VE (π.χ $conda install pycrypto) ή μέσω του Python Package Manager (pip)

   d. Προφανώς ο κώδικας σας μπορεί να τρέξει και σε prompt γράφοντας: $python "source.py"

2. Εγκαταστήστε στο VE Crypto libs όπου χρειάζεται:
   a. https://pypi.python.org/pypi/pycrypto
   b. https://pypi.python.org/pypi/ssl