1η προαιρετική εργασία

Θεμελιώσεις Κρυπτογραφίας

Αναγνώστου Αντώνιος – 2268 – [anagnoad@csd.auth.gr](mailto:anagnoad@csd.auth.gr)

Λασκαρίδησ Στέφανος – 2315 – [laskstef@csd.auth.gr](mailto:Laskstef@csd.auth.gr)

2015

Περιεχόμενα

[Εισαγωγή 2](#_Toc416444989)

[Θέμα 1ο 2](#_Toc416444990)

[Υποερωτήμα i) 2](#_Toc416444991)

[Υποερώτημα ii) 2](#_Toc416444992)

[Υποερώτημα iii) 2](#_Toc416444993)

[Υποερώτημα iv) 3](#_Toc416444994)

[Θέμα 2ο 3](#_Toc416444995)

[Θέμα 3ο 3](#_Toc416444996)

[Θέμα 4ο 3](#_Toc416444997)

[Θέμα 5ο 3](#_Toc416444998)

[Θέμα 6ο 4](#_Toc416444999)

[Θέμα 7ο 4](#_Toc416445000)

[Θέμα 8ο 4](#_Toc416445001)

[Θέμα 9ο 4](#_Toc416445002)

[Υποερώτημα i) 4](#_Toc416445003)

[Υποερώτημα ii) 5](#_Toc416445004)

[References 6](#_Toc416445005)

# Εισαγωγή

Η παρούσα εργασία αποσκοπεί στην εξοικίωση με τις βασικές έννοιες της κρυπτογραφίας, εξάσκηση με τους αλγόριθμους κρυπτογράφησης καθώς και στην εφαρμογή μεθόδων κρυπτάναλυσης για την παραβίαση κρυπτοσυστημάτων.

Για όλες τις προγραμματιστικές ασκήσεις που ακολουθούν, χρησιμοποιήθηκαν τα ακόλουθα:

* Python 3
* pycrypt library
* Sage math software
* OS: Ubuntu 14.04, 64-bit
* Github repository: https://github.com/stevelaskaridis/cryptography\_project1

# Θέμα 1ο

## Υποερωτήμα i)

Αρχή Kerchoff:

Σύμφωνα με την αρχή του Kerkhoff, η ασφάλεια ενός κρυπτοσυστήματος θα πρέπει να βασίζεται εξ'ολοκλήρου στη μυστικότητα του κλειδιού και στην τυχαιότητα των γεννητριών τυχαίων bitstrings. Αυτό σημαίνει πως η μυστικότητα του κώδικα και της υλοποίησης του κρυπτοσυστήματος δεν θα πρέπει να προσδίδει το στοιχείο της ασφάλειας στο σύστημά μας.

Ακόμη και αν ο επιτιθέμενος γνωρίζει τους αλγορίθμους υλοποίησης του συστήματος, δε θα πρέπει να μπορεί να "σπάσει" το κρυπτοσύστημα λόγω αυτής του της γνώσης.

Πλεονεκτήματα αυτού του κανόνα αποτελούν:

1. Ο κώδικας υλοποίησης του κρυπτοσυστήματος μπορεί να είναι open-source, με συνέπεια την παρακολούθηση και εξέτασή του από μεγαλύτερη μερίδα προγραμματιστών. ("Every bug is shallow, given enough eyes")

2. Ακόμη και αν οι γνώστες της υλοποίησης του κρυπτοσυστήματος αποτελούν τμήμα της επίθεσης προς το σύστημα, η γνώση δεν προσδίδει καμία πληροφορία σε σχέση με το κλειδί.

3. Στα πλαίσια ανοχής παραβιάσεων, αν η λειτουργία και ασφάλεια του συστήματος βασίζεται στη μυστικότητα του αλγορίοθμου υλοποίησης, αποτελεί ενδεχόμενο failure point.

## Υποερώτημα ii)

http://pileas.csd.auth.gr/file.php/141/cource-1-2-3.pdf [94]

Κρυπτοσυστήματα με τέλεια ασφάλεια:

Vigénere, OTP

<http://crypto.stackexchange.com/questions/3896/simply-put-what-does-perfect-secrecy-means>

## Υποερώτημα iii)

Όχι. Στην περίπτωση που κρυπτογραφηθούν δύο μηνύματα m1, m2 με το ίδιο κλειδί k, και τα αντίστοιχα ciphertexts είναι c1 και c2, τότε ισχύει το παρακάτω:

c1 xor c2 = (m1 xor k) xor (m2 xor k) = m1 xor m2

Αν τα μηνύματα είναι σε φυσική γλώσσα και ακολουθείται κάποια κατανομή χρήσης των γραμμάτων, τότε μπορούν να βρεθούν τα μηνύματα. Τα μηνύματα, ωστόσο, πρέπει να έχουν πλεονασμό.

## Υποερώτημα iv)

Η κατάσταση λειτουργίας σε ένα κρυπτοσύστημα τμήματος αναφέρεται στο πώς χρησιμοποιείται επαναληπτικά ο αλγόριθμος κρυπτογράφησης τμήματος για να μετασχηματίσει τμήματα τα οποία είναι μεγαλύτερα από το τμήμα (?)

Κατάσταση λειτουργίας ECB (Electronic Codebook)

Η απλούστερη κατάσταση λειτουργίας. Το μήνυμα διαιρείται σε blocks, καθένα από τα οποία κρυπτογραφείται αυτόνομα.

Το μειονέκτημα αυτής της μεθόδου είναι ότι επιτρέπει την ανίχνευση patterns στο κρυπτογραφημένο ciphertext, καθώς δεδομένου ότι παρόμοια blocks του μηνύματος κρυπτογραφούνται με το ίδιο κλειδί, έχουν και παρόμοιο παραγόμενο ciphertext.

# Θέμα 2ο

# Θέμα 3ο

# Θέμα 4ο

Το μήνυμα είναι το ακόλουθο:

«ΜΗΔΕΙΣ ΑΓΕΩΜΕΤΡΗΤΟΣ ΕΙΣΙΤΩ ΜΟΥ ΤΗΝ ΣΤΕΓΗΝ»

# Θέμα 5ο

Υπάρχει "bug" στη βιβλιοθήκη του zipfile όπου με διαφορετικό κωδικό του σωστού, μπορεί να γίνει extract του αρχείου με μηδενικό μέγεθος.

Στην προτεινόμενη λύση, αγνοούνται όλα τα exceptions και γίνονται οι απαραίτητοι έλεγχοι για το πότε το extraction έχει πραγματοποιηθεί επιτυχώς.

# Θέμα 6ο

# Θέμα 7ο

# Θέμα 8ο

# Θέμα 9ο

## Υποερώτημα i)

Ο αλγόριθμος Blowfish είναι ένας συμμετρικός αλγόριθμος κρυπtογράφησης τμήματος που προτάθηκε το 1993 από τον Bruce Scneier και έχει αρκετά μεγάλη διάδοση στη σημερινή εποχή. Παρέχει αποδεκτό ρυθμό κρυπτογράφησης και δεν έχει υπάρξει κάποια μέθοδος κρυπτανάλυσης που να έχει δημοσιευτεί μέχρι σήμερα. Αποτελεί εναλλακτική στο δημοφιλέστατο AES (Advanced Encryption Standard), με τον δεύτερο να έχει μεγαλύτερο μερίδιο χρήσης.

Γενικά, ο Blowfish θεωρείται πως είναι ένας αρκετά γρήγορος αλγόριθμος κρυπτογράφησης, εκτός από την περίπτωση αλλαγής κλειδιών. Κάθε νέο κλειδί χρειάζεται μία προεπεξεργασία αντίστοιχη με την κρυπτογράφηση κειμένου 4KB, το οποίο θεωρείται εξαιρετικά αργό σε σύγκριση με άλλους αλγοριίθμους. Ωστόσο, το γεγονός αυτό θα μπορούσε να λειτουργήσει και θετικά: για παράδειγμα, το OpenBSD χρησιμοποιεί έναν αλγόριθμο εμπνευσμένο από το Blowfish που αξιοποιεί αυτή την καθυστέρηση, για να αμυνθεί απέναντη σε dictionary attacks. Άλλες μετεξελίξεις του BlowFish αποτελούν οι αλγόριμοι Twofish και Threefish.

Περιγραφή λειτουργίας:

Ο αλγόριθμος, όπως ισχύει και στον DES και τον AES, κάνει χρήση S-boxes. Πιο συγκεκριμένα, ο Blowfish έχει block μεγέθους 64-bit και χρησιμοποιεί το σχήμα Feistel 16 φάσεων. Το μήκος του κλειδιού μεταβάλλεται από 32 μέχρι 448 bits, ενώ χρησιμοποιεί μεγάλα S-boxes που εξαρτώνται από το κλειδί.

Αναλυτικότερα, ο αλγόριθμος χρησιμοποιεί δύο subkey arrays: ένα array P 18 θέσεων, καθώς και 4 S-boxes των 256-bit. Τα S-boxes έχουν εισόδους των 8 bit ενώ το παραγόμενο αποτέλεσμα είναι 32 bit. Σε κάθε φάση χρησιμοποιείται μία μόνο θέση από το P και μετά τον τελευταίο γύρο, κάθε data block χωρίζεται στη μέση και πραγματοποιείται η πράξη XOR με μία από τις δύο εναπομείναντες θέσεις του πίνακα P.

Έχοντας πει τα παραπάνω, μπορούμε να περιγράψουμε και την συνάρτηση F του Blowfish. Η συνάρτηση αυτή χωρίζει μία είσοδο 32-bit σε 4 τμήματα των 8 bits και τοποθετεί τα bits αυτά σαν είσοδο στα S-boxes. Οι έξοδοι προστίθενται με αριθμητική modulo 2^32 και αφού πραγματοποιηθεί λογική αποκλειστική διάζευξη (XOR) παράγεται το τελικό output των 32-bit.

Η αποκρυπτογράφηση λειτουργεί με αντίστοιχο τρόπο, με την μόνη διαφορά πως οι 18 θέσεις του P (P1, P2, ..., P18) βρίσκονται σε αντίστροφη σειρά.

Όσον αφορά στην διαδικασία παραγωγής των subkeys, ο Blowfish αρχικοποιεί το P και τα S-boxes με τις δεκαεξαδικές τιμές των ψηφίων του άρρητου αριθμού π, αφού δεν υπάρχει κάποια αλληλουχία στους αριθμούς αυτούς. Έπειτα πραγματοποιείται XOR του κάθε byte του κλειδιού (επαναλμβάνοντας αυτό αν χρειαστεί), με τις θέσεις του P. Στη συνέχεια, κρυπτογραφείται ένα μηδενικό block 64 bit και το παραγόμενο κρυπτοκείμενο αντικαθιστά τα P1 και P2. Το ίδιο κρπυτοκείμενο κρυπτογραφείται ξανά, με νέα subkeys, και το νέο output αντικαθιστά τα P3 και P4. Η διαδικασία αυτή συνεχίζει, αντικαθιστώντας ολόκληρο το array P και τα S-boxes. Συνολικά δυλαδή, ο αλγόριθμος Blowfish χρειάζεται 521 περάσματα για να παράξει όλα τα subkeys.

Τρωτά σημεία:

Ο Blowfish θεωρείται ευάλωτος σε επιθέσεις όπου χρησιμοποιούνται "αδύναμα" κλειδιά. Οι χρήστες θα πρέπει να αποφεύγουν ορισμένες κλάσεις κλειδιών που είναι γνωστές για την αδυναμίες τους στη χρήση τους με τον Blowfish ή να στραφούν προς εναλλακτικές όπως ο AES, ο ChaCha (που χρησιμοποιείται από τη Google) ή ο Salsa20.

## Υποερώτημα ii)

# References

**There are no sources in the current document.**