Ατομική Διπλωματική Εργασία

ΕΡΕΥΝΑ ΑΣΦΑΛΕΙΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΕΜΦΑΣΗ ΣΤΟΝ ΑΛΓΟΡΙΘΜΟ ΚΡΥΠΤΟΓΡΑΦΙΣΗΣ ELLIPTIC CURVES

Χριστίνα Προκοπίου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Έρευνα ασφάλειας συστημάτων με έμφαση στον αλγόριθμο κρυπτογράφησης Elliptic Curves

Χριστίνα Προκοπίου

Επιβλέπων Καθηγητής

Δρ. Ηλίας Αθανασόπουλος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2020

Ευχαριστίες

Θα ήθελα να εκμεταλλευτώ την ευκαιρία να εκφράσω την εκτίμησή μου προς τον Επιβλέπων Καθηγητή και Ακαδημαϊκό μου σύμβουλο, Δρ. Ηλία Αθανασόπουλο, για τις ανεκτίμητες συμβουλές και καθοδήγηση του κατά τη διάρκεια αυτής της διπλωματικής εργασίας. Η συνεχής βοήθειά του ήταν κρίσιμη για να ξεπεράσω όλες τις δυσκολίες που έχω αντιμετωπίσει σε αυτή την έρευνα. Τις βαθύτατες μου ευχαριστίες.

Τέλος, δεν θα μπορούσα να μην αναφέρω την οικογένεια μου που ήταν δίπλα μου σε κάθε στάδιο και επέδειξαν κατανόηση και συμπαράσταση κατά την διάρκεια αυτής της απαιτητικής περιόδου.

Περίληψη

Τα τελευταία χρόνια έχει γίνει μεγάλη έρευνα σχετικά με την τυχαιότητα και τη δημιουργία ψευδοτυχαίων string of bits. Ο σκοπός αυτής της μελέτης είναι να εξετάσει την τυχαιότητα της δημιουργίας κλειδιών χρησιμοποιώντας τον αλγόριθμο κρυπτογράφησης Elliptic Curves και να εμβαθύνει στο πόσο τυχαία είναι αυτά τα κλειδιά. Υπάρχουν πολλές βιβλιοθήκες κρυπτογράφησης οι οποίες μπορούν να συμβάλουν στην παραγωγή ζεύγους κλειδιών Elliptic Curves. Για σκοπούς αυτής της διπλωματικής εργασίας όμως έχουν μελετηθεί οι βιβλιοθήκες OpenSSL, LibSodium/NaCl και BouncyCastle. Ο στόχος είναι να γίνουν δοκιμές τυχαιοποίησης στο κάθε ζεύγος κλειδιών που δημιουργούνται από κάθε βιβλιοθήκη και να γίνει μια εξονυχιστική ανάλυση τόσο για την προέλευση ενός κλειδιού όσο και για την τυχαιότητα του. Ως εκ τούτου, αυτή η έρευνα ίσως δώσει μια απάντηση στο ερώτημα εάν υπάρχουν μοτίβα κατά τη δημιουργία των Elliptic Curve κλειδιών. Εάν η απάντηση στην προηγούμενη ερώτηση είναι θετική δηλαδή εάν κάποιος μπορεί να διαπιστώσει την προέλευση του κλειδιού ή να μπορέσει να σπάσει την τυχαιότητα του, αυτό καθιστά τον αλγόριθμο ευάλωτο σε επιθέσεις καθώς επίσης οδηγεί στην έλλειψη ασφάλειας. Οι προαναφερθέντες κίνδυνοι μπορούν εύκολα να παρουσιαστούν αν από ένα κλειδί διαρρεύσουν πληροφορίες για τις επιλογές σχεδιασμού και υλοποίησης όπως τον πρωταρχικό αλγόριθμος παραγωγής του.

Η ψηφιακή κρυπτογραφία βασίζεται σε μεγάλο βαθμό στην τυχαιότητα και στην παροχή ασφαλείας που επιβάλλονται από διάφορα συστήματα πληροφοριών. Απαιτείται η τυχαιότητα σε μια ποικιλία ρόλων προκειμένου να διασφαλιστεί η σωστή δύναμη των κρυπτογραφικών τεχνικών. Ο σκοπός της έρευνας είναι να τονίσει τη σημασία της τυχαιότητας στην ψηφιακή κρυπτογραφία για να επιδείξει τη σημασία της επιλογής και της ενσωμάτωσης κατάλληλων γεννητριών τυχαίων αριθμών γιατί ένα ελαττώματα ασφαλείας στη γεννήτρια μπορεί εύκολα να θέσει σε κίνδυνο την ασφάλεια ολόκληρου του συστήματος.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Οι αλγόριθμοι κρυπτογράφησης εφαρμόζονται σε έναν αυξανόμενο αριθμό συσκευών για να ικανοποιήσουν τις υψηλές απαιτήσεις ασφαλείας τους. Πολλές από αυτές τις συσκευές απαιτούν λειτουργία υψηλής ταχύτητας και περιλαμβάνουν εξειδικευμένα κυκλώματα κρυπτογράφησης και αποκρυπτογράφησης υλικού. Ένα μοναδικό χαρακτηριστικό αυτών των κυκλωμάτων είναι η πολύ υψηλή ευαισθησία τους σε σφάλματα. Σε αντίθεση με τα συνηθισμένα αριθμητικά ή λογικά κυκλώματα όπως είναι οι adders και οι multipliers, ακόμη και ένα σφάλμα data bit σε ένα κύκλωμα κρυπτογράφησης ή αποκρυπτογράφησης, στις περισσότερες περιπτώσεις, θα εξαπλωθεί γρήγορα και θα οδηγήσει σε μια τελείως λανθασμένη έξοδο. Υπάρχει, επομένως, ανάγκη να αποφευχθούν τέτοια σφάλματα ή, τουλάχιστον, να είναι δυνατή η ανίχνευσή τους.

Πολλοί κρυπτογραφικοί αλγόριθμοι βασίζονται σε ισχυρούς ψευδο-τυχαίους αριθμούς για να λειτουργούν με ασφάλεια. Κάθε καλή βιβλιοθήκη που παρέχει αλγόριθμους κρυπτογράφησης και κατακερματισμού θα παρέχει επίσης οδηγίες για τη δημιουργία τυχαίων αριθμών.

Ένα κοινό λάθος σχετικά με τη δημιουργία τυχαίων αριθμών (δηλ. Δημιουργία εντροπίας) είναι ο συσχετισμός της ποσότητας των bit με την προβλεψιμότητα των bit. Για παράδειγμα, ένας ακέραιος 32-bit που δημιουργείται από τη συνάρτηση rand () ενός συστήματος που στη συνέχεια κατακερματίζεται από τον SHA-256 για να δημιουργήσει μια τιμή 256-bit δεν έχει γίνει "πιο τυχαίος". Η πηγή της εντροπίας παραμένει τόσο καλή όσο η πηγή που χρησιμοποιείται από το rand () για τη δημιουργία του αρχικού ακέραιου. Υποθέτοντας ότι ο ακέραιος αριθμός 32-bit κατανέμεται ομοιόμορφα στο πιθανό εύρος, τότε ένας εισβολέας πρέπει να στοχεύσει χώρο 32-bit και όχι χώρο 256-bit.

Η ασφάλεια των συστημάτων κρυπτογράφησης σχετίζεται στενά με την τυχαιότητα για αυτό το λόγο τα τελευταία χρόνια έχουν γίνει πολλές έρευνες για την τυχαιότητα διαφόρων αλγορίθμων κρυπτογράφησης. Ο σκοπός αυτής της έρευνας λοιπόν είναι να εξετάσει την τυχαιότητα της δημιουργίας κλειδιών χρησιμοποιώντας τον αλγόριθμο κρυπτογράφησης Elliptic Curves και να εμβαθύνει στο πόσο πραγματικά τυχαία είναι αυτά τα κλειδιά.

Κεφάλαιο 2

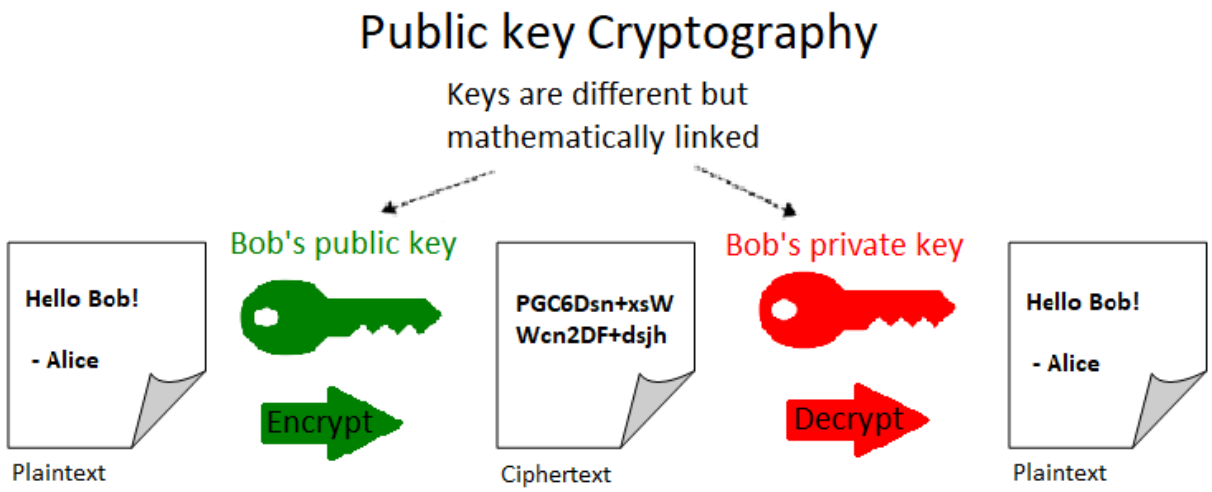
Υπόβαθρο

2.1 Κρυπτογράφηση κλειδιών

Στην κρυπτογραφία, ένα κλειδί είναι ένα κομμάτι πληροφορίας (μια παράμετρος) που καθορίζει τη λειτουργική έξοδο ενός κρυπτογραφικού αλγορίθμου. Για τους αλγόριθμους κρυπτογράφησης, ένα κλειδί καθορίζει τη μετατροπή του απλού κειμένου σε κρυπτοκείμενο, και το αντίστροφο για αλγόριθμους αποκρυπτογράφησης. Τα κλειδιά καθορίζουν επίσης μετασχηματισμούς σε άλλους κρυπτογραφικούς αλγόριθμους, όπως σχήματα ψηφιακής υπογραφής και κωδικούς ελέγχου ταυτότητας μηνυμάτων [1].

Τα κλειδιά που χρησιμοποιούνται στην public key κρυπτογραφία έχουν μαθηματική δομή. Για παράδειγμα, τα δημόσια κλειδιά που χρησιμοποιούνται στο σύστημα RSA είναι το προϊόν δύο πρώτων αριθμών. Έτσι, τα public key συστήματα απαιτούν μεγαλύτερα μήκη κλειδιών από τα συμμετρικά συστήματα για ισοδύναμο επίπεδο ασφάλειας. 3072 bits είναι το προτεινόμενο μήκος κλειδιού για συστήματα που βασίζονται σε factoring και ακέραιους διακριτούς λογάριθμους που στοχεύουν στην ασφάλεια ισοδύναμη με την συμμετρική κρυπτογράφηση 128 bit. Η Elliptic Curve κρυπτογραφία μπορεί να επιτρέπει κλειδιά μικρότερου μεγέθους για ισοδύναμη ασφάλεια. Ο τρέχων κανόνας είναι να χρησιμοποιείται ένα κλειδί ECC δύο φορές περισσότερο από το επιθυμητό επίπεδο ασφαλείας του συμμετρικού κλειδιού.

Τα σύγχρονα κρυπτογραφικά συστήματα περιλαμβάνουν αλγόριθμους συμμετρικού κλειδιού (όπως DES και AES) και αλγόριθμους public key (όπως RSA). Οι αλγόριθμοι συμμετρικού κλειδιού χρησιμοποιούν ένα κοινόχρηστο κλειδί και η διατήρηση μυστικών δεδομένων απαιτεί τη διατήρηση αυτού του κλειδιού μυστικού. Οι αλγόριθμοι public key χρησιμοποιούν public key και private key. Το public key διατίθεται σε όλους (συχνά μέσω ψηφιακού πιστοποιητικού). Ένας αποστολέας κρυπτογραφεί δεδομένα με το public key του παραλήπτη και μόνο ο κάτοχος του private key μπορεί να αποκρυπτογραφήσει αυτά τα δεδομένα.



Εικόνα 2.1 Η εικόνα αυτή απεικονίζει την επικοινωνία με την χρήση public και private keys

Η κρυπτογραφία υπολογιστή χρησιμοποιεί ακέραιους αριθμούς για κλειδιά. Σε ορισμένες περιπτώσεις, τα κλειδιά δημιουργούνται τυχαία χρησιμοποιώντας μια γεννήτρια τυχαίων αριθμών (RNG – Random Number Generator) ή μια ψευδοτυχαία γεννήτρια αριθμών (PRNG – Pseudo Random Number Generator). Το PRNG είναι ένας αλγόριθμος που παράγει δεδομένα που εμφανίζονται τυχαία υπό ανάλυση. Τα PRNG που χρησιμοποιούν εντροπία συστήματος γενικά παράγουν καλύτερα αποτελέσματα, καθώς αυτό καθιστά τις αρχικές συνθήκες του PRNG πολύ πιο δύσκολες για έναν εισβολέα να μαντέψει. Ένας άλλος τρόπος δημιουργίας τυχαιότητας είναι η χρήση πληροφοριών εκτός του συστήματος. Το veracrypt (ένα λογισμικό κρυπτογράφησης δίσκου) χρησιμοποιεί κινήσεις ποντικιού χρήστη για τη δημιουργία μοναδικών seeds, στους οποίους οι χρήστες ενθαρρύνονται να μετακινούν το ποντίκι τους τακτικά.

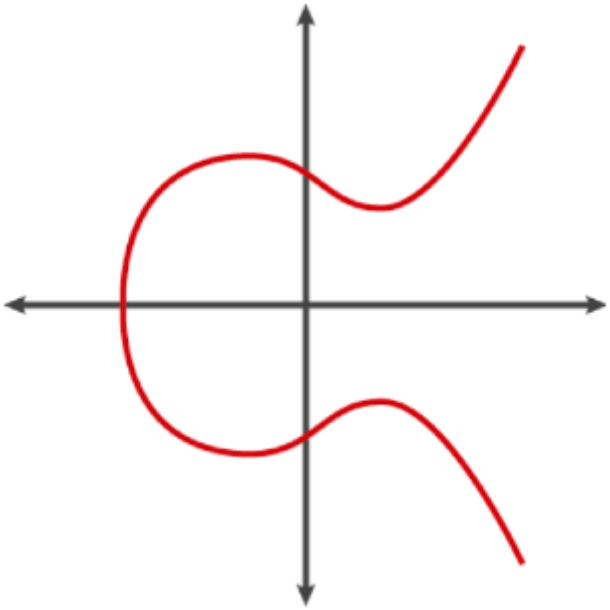
Η απλούστερη μέθοδος ανάγνωσης κρυπτογραφημένων δεδομένων χωρίς την πραγματική αποκρυπτογράφηση είναι μια επίθεση ωμής βίας (brute-force attack). Επομένως, είναι σημαντικό να χρησιμοποιείται ένα αρκετά μεγάλο μήκος κλειδιού. Τα μεγαλύτερα σε μήκος κλειδιά χρειάζονται εκθετικά περισσότερο χρόνο για να επιτεθούν, καθιστώντας την επίθεση ωμής βίας μη πρακτική.

2.2 Ο αλγόριθμος Elliptic Curves

Η Elliptic Curves Cryptography (ECC) [2] είναι μια προσέγγιση στην κρυπτογραφία public key που βασίζεται στην αλγεβρική δομή των ελλειπτικών καμπυλών πάνω σε πεπερασμένα πεδία. Η ECC απαιτεί μικρότερα κλειδιά σε σύγκριση με την κρυπτογραφία εκτός EC (βάσει απλών πεδίων Galois) για την παροχή ισοδύναμης ασφάλειας. Η ασφάλεια της ECC βασίζεται την αδυναμία υπολογισμού του πολλαπλασιαστέου που δίνεται και των σημείων του προϊόντος. Επίσης, το σημαντικό όφελος που παρέχει η ECC είναι η δυνατότητα χρήσης ενός μικρότερου κλειδιού, αλλά διατηρεί ένα ισχυρό επίπεδο ασφάλεια. Η χρήση του ECC είναι ένας τρόπος δημιουργίας γρηγορότερων, μικρότερων και αποδοτικότερων κλειδιών κρυπτογραφίας.

2.1.1 Επεξήγηση ελλειπτικών καμπυλών:

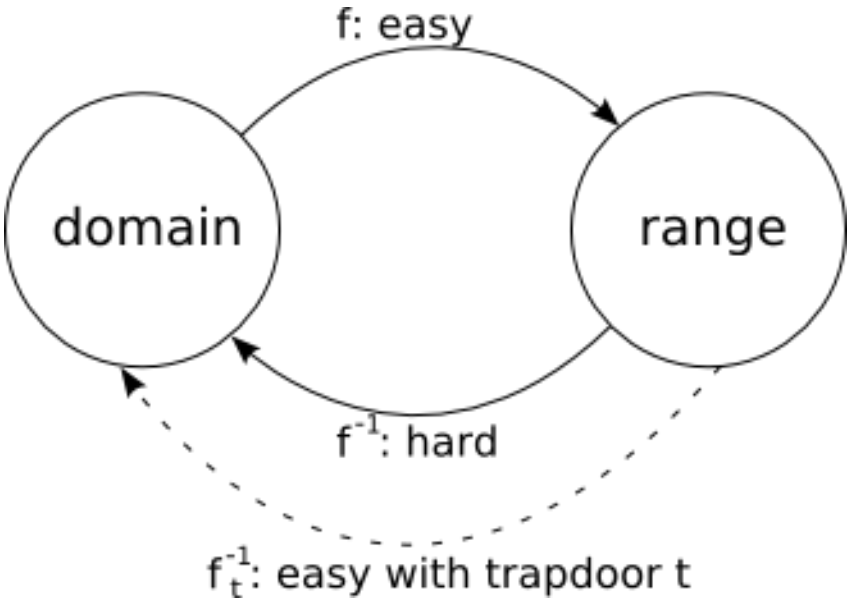
Μια ελλειπτική καμπύλη είναι το σύνολο των σημείων που ικανοποιούν μια συγκεκριμένη μαθηματική εξίσωση. Η εξίσωση για μια ελλειπτική καμπύλη μοιάζει με $y^2 = x^3 + ax + b$ και αντιπροσωπεύεται γραφικά όπως στην Εικόνα 2.2.



Εικόνα 2.2 Παράδειγμα ελλειπτικής καμπύλης

Όλα τα κρυπτοσυστήματα χρησιμοποιούν μια λειτουργία την λεγόμενη Trapdoor function για την εφαρμογή και την ασφάλειά τους. Η λειτουργία Trapdoor [3] είναι μια λειτουργία που είναι εύκολο να υπολογιστεί με έναν τρόπο, αλλά είναι πολύ δύσκολο, μαθηματικά ανέφικτο, ο υπολογισμό της αντίστροφής κατεύθυνσής της. Για παράδειγμα, από το Α πάω στο Β πολύ εύκολα αλλά από το Β να επιστρέψω στο Α είναι μαθηματικά αδύνατο.

Οι λειτουργίες Trapdoor έχουν μια παρόμοια αρχή με τις μονόδρομες λειτουργίες. Ωστόσο, στις λειτουργίες Trapdoor υπάρχει ένα ειδικό κλειδί, το οποίο εάν είναι γνωστό, είναι δυνατόν και εύκολο να αντιστραφεί και να υπολογιστεί η αντίστροφη κατεύθυνση. Από την άλλη πλευρά, η μονόδρομη λειτουργία δεν είναι αναστρέψιμη, επειδή δεν υπάρχει επιστροφή πίσω. Με άλλα λόγια, μια λειτουργία Trapdoor, όπως υποδηλώνει το όνομα, καθιστά εύκολο να βρεθείς σε μια παγίδα, αλλά είναι πολύ δύσκολο να ξεφύγεις από αυτήν, εκτός αν έχεις τις ειδικές γνώσεις ενός κλειδιού.



Εικόνα 2.3 Μαθηματική εξήγηση του Trapdoor Function

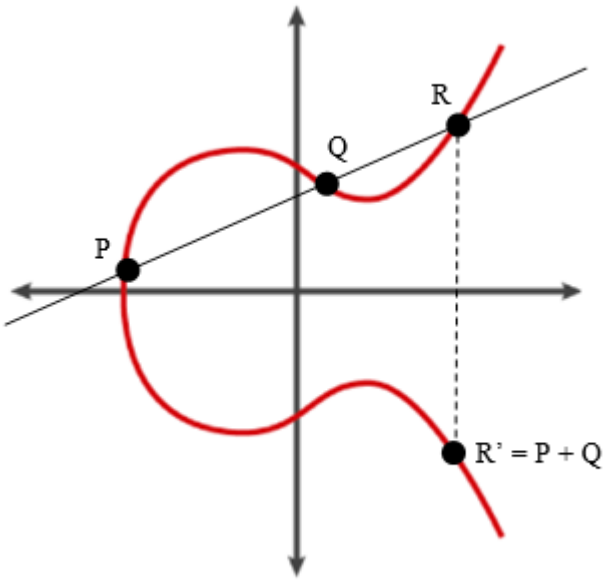
Για παράδειγμα, ο RSA χρησιμοποιεί μια λειτουργία Trapdoor που βασίζεται στην έννοια του Prime Factorization. Πιο συγκεκριμένα στον RSA είναι εύκολο να υπολογιστεί το n εάν τα p και q είναι γνωστά, n = p * q. Ωστόσο, ο αντίστροφος υπολογισμός είναι μαθηματικά δύσκολος. Η ειδική γνώση για τον αντίστροφο υπολογισμό είναι το p ή το q ή και τα δύο. Είναι εξαιρετικά δύσκολο να υπολογιστεί το p και το q από το factoring n, αφού τα p και q είναι μεγάλοι πρώτοι αριθμοί (1024 bit).

Παρομοίως, η ECC χρησιμοποιεί μια λειτουργία Trapdoor, αλλά σε αυτή την περίπτωση η συνάρτηση βασίζεται σε πολλαπλασιασμό σημείων. Η εξίσωση που παρουσιάζεται ως συνάρτηση Trapdoor του ECC είναι A = n * B, στην οποία A είναι το τελικό σημείο που παράγεται πολλαπλασιάζοντας το σημείο εκκίνησης B με το n. Είναι απλό και εύκολο να υπολογιστεί το A, η μία κατεύθυνση της εξίσωσης, αλλά είναι δύσκολο να βρεθεί το n όταν είναι γνωστά και τα δύο A και B. Η ειδική γνώση που απαιτείται για τον αντίστροφο υπολογισμό είναι το n όπου $n \in Z$.

2.1.2 Εξισώσεις ελλειπτικών καμπύλων:

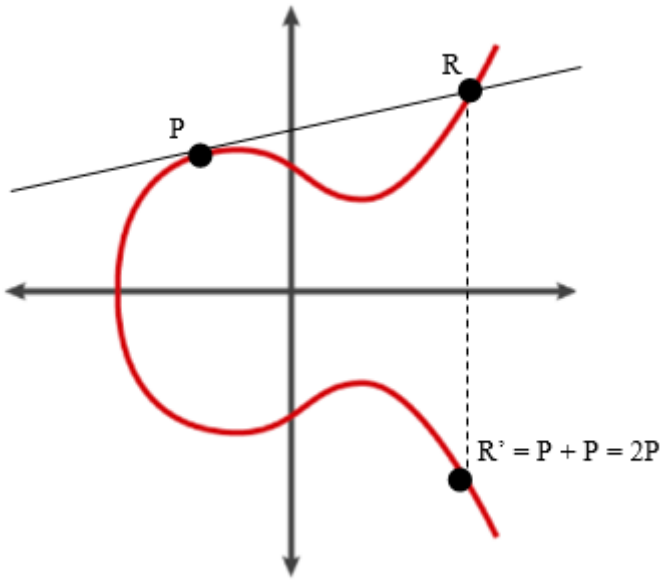
Τα κρυπτοσυστήματα EC βασίζονται σε εξισώσεις ελλειπτικής καμπύλης [4]. Υπάρχουν πολλές εξισώσεις, επομένως κάθε κρυπτοσύστημα επιλέγει μία. Ωστόσο, τα σημεία που ισχύουν για την κάθε εξίσωση είναι σημεία πάνω στην καμπύλη. Γενικά, τα κρυπτοσυστήματα που βασίζονται στο ECC χρησιμοποιούν τη εξίσωση, $y^2 = x^3 + ax + b$.

Πάντα, δύο σημεία σε μια ΕC τέμνουν ένα άλλο τρίτο σημείο, εκτός από τα σημεία που είναι κατακόρυφα. Για παράδειγμα, εάν έχουμε ένα σημείο P και ένα σημείο Q σε μια ΕC τότε δημιουργείται ένα τρίτο σημείο R. Υποθέτοντας ότι πρέπει να προσθέσουμε το σημείο P με το σημείο Q, τότε αντικατοπτρίζουμε το τρίτο σημείο R για να πάρουμε το σημείο R'. Το ανακλώμενο σημείο R' είναι το αποτέλεσμα της προσθήκης του σημείου P και το σημείο Q. Η διαδικασία που έχει περιγράφει ονομάζεται Point Addition και φαίνεται στην Εικόνα 2.4.



Εικόνα 2.4 Point addition

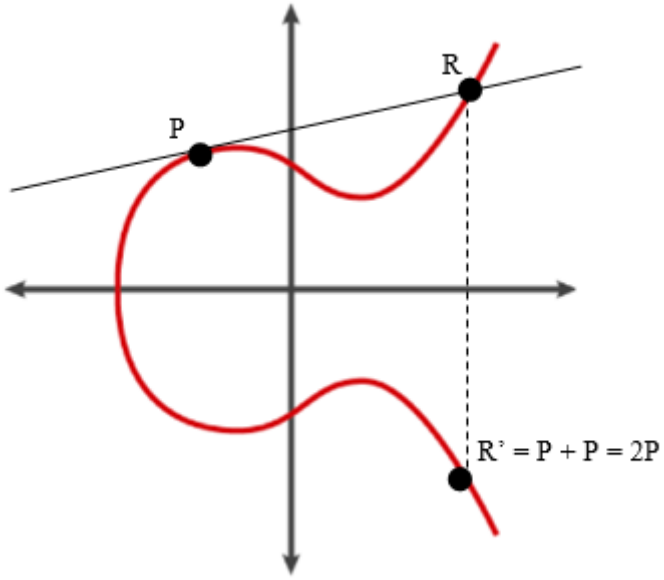
Από την άλλη πλευρά, μπορεί ένα σημείο να προστεθεί με τον εαυτό του. Αυτή η μέθοδος σε ΕC ονομάζεται Point Doubling. Στο Σημείο Διπλασιασμού δεν υπάρχει δεύτερο σημείο, επομένως δεν μπορούμε να δημιουργήσουμε το τρίτο σημείο. Αντί αυτού, σχεδιάζουμε την εφαπτομένη του σημείου P και έπειτα παίρνουμε το τέμνον σημείο του να είναι το R. Παρομοίως με το Point Addition, αντικατοπτρίζουμε το R για να πάρουμε το σημείο R' που είναι το αποτέλεσμα του Point Doubling, R' = P + P = 2P . Αυτό φαίνεται στην Εικόνα 2.5.



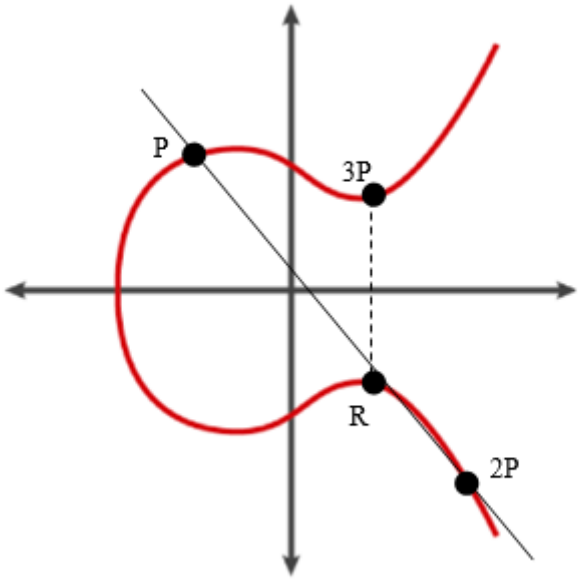
Εικόνα 2.5 Point Doubling

Όπως φαίνεται στην Εικόνα 2.5 το σημείο R είναι κατακόρυφο στο σημείο R', οπότε δεν εφαρμόζουμε ούτε Point Addition ή Point Doubling. Η προσθήκη δύο κάθετων σημείων είναι μια απροσδιόριστη διαδικασία. Εάν προσπαθήσουμε να προσθέσουμε δύο κάθετα σημεία, δεν θα υπάρξει ποτέ ένα τρίτο σημείο και το αποτέλεσμα της προσθήκης δύο κατακόρυφων σημείων θα είναι το άπειρο.

Επιπλέον, η λειτουργία πολλαπλασιασμού σημείων βασίζεται στον τρόπο λειτουργίας της Point Addition και Point Doubling. Στο πολλαπλασιασμό σημείων, η λειτουργία του Point Addition εφαρμόζεται k φορές, όπου το k είναι ο παράγοντας του πολλαπλασιασμού. Για παράδειγμα, όταν πρέπει να πολλαπλασιάσουμε το σημείο P τρεις φορές, 3P = P + P + P = 2P + P, πρέπει να εκτελέσουμε διαδοχικά Point Doubling και Point Addition. Το παράδειγμα για τον υπολογισμό του 3P φαίνεται στις Εικόνες 2.6 και 2.7. Αρχικά, το σημείο που αντιστοιχεί στο 2P υπολογίζεται ως 2P = P + P χρησιμοποιώντας δηλαδή το Point Doubling. Αφού το σημείο P είναι γνωστό, χρησιμοποιώντας την εφαπτομένη μπορούμε να βρούμε το σημείο 2P. Για το δεύτερο βήμα, δεδομένου ότι είναι γνωστά τα σημεία 2P και P, η λειτουργία Point Addition μπορεί να χρησιμοποιηθεί για τον υπολογισμό του σημείου που αντιστοιχεί σε 3P, 3P = 2P + P.

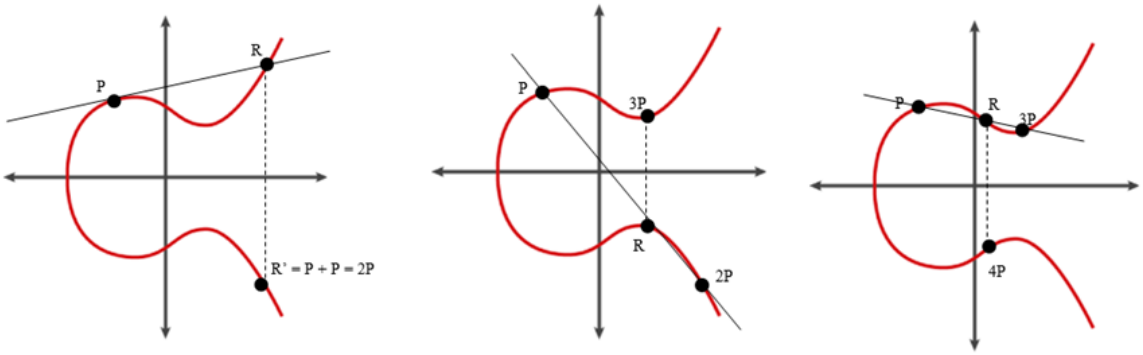


Εικόνα 2.6



Εικόνα 2.7

Ένα άλλο παράδειγμα είναι το τετραπλάσιο του σημείου P. Τα βήματα του παραδείγματος παρουσιάζονται στην εικόνα 2.8. Ο πολλαπλασιασμός σημείων για τον υπολογισμό του 4P είναι συνέχεια του προηγούμενου παραδείγματος, στο οποίο το σημείο που υπολογίζεται αντιστοιχεί στο 3P. Δεδομένου ότι έχουμε και τα δύο σημεία 3P και P, η λειτουργία Point Addition μπορεί να χρησιμοποιηθεί για τον υπολογισμό των 4P δηλαδή 4P = 3P + P. Επιπλέον, το 4P μπορεί να υπολογιστεί χρησιμοποιώντας τη λειτουργία Point Doubling δηλαδή 4P = 2P + 2P.



Εικόνα 2.8 Παράδειγμα με 4P

2.3 Παραγωγή ζεύγους EC κλειδιών

Οι ελλειπτικές καμπύλες βασίζονται στο πρόβλημα Discrete Logarithm (DL). Στα κρυπτοσυστήματα που χρησιμοποιούν EC, το πρόβλημα DL βοηθά στη διαδικασία δημιουργίας κλειδιών. Εάν υποθέσουμε ότι έχουμε μια εξίσωση, που παρουσιάζει ένα EC που ονομάζεται E, ένα σημείο P στο EC E και ένα άλλο σημείο στο E το T, όπου το σημείο T είναι το αποτέλεσμα της απόδοσης του πολλαπλασιασμού σημείων ECs στο σημείο P, δηλαδή T = dP = P + P + P ... + P. Στα κρυπτοσυστήματα EC, το T είναι το public key και το d είναι το private key. Η παραπάνω περιγραφή ονομάζεται Elliptic Curved Discrete Logarithm Problem (ECDLP), αλλά το d υπολογίζεται από το πρόβλημα DL.

Η ανταλλαγή κλειδιών με ελλειπτικές καμπύλες ονομάζεται ανταλλαγή κλειδιών Elliptic Curve Diffie-Hellman (ECDH). Στο ECDH, είναι απαραίτητο να προσδιοριστούν οι παράμετροι τομέα (Domain parameters). Οι παράμετροι τομέα είναι ένας πρώτος αριθμός p, η ελλειπτική καμπύλη E και ένα αρχικό στοιχείο P που είναι ένα σημείο στην καμπύλη E. Η δυσκολία στον προσδιορισμό των παραμέτρων τομέα είναι να βρεθεί η ελλειπτική καμπύλη που ικανοποιεί τις ιδιότητες ασφαλείας του κρυπτοσυστήματος.

Για παράδειγμα, η Alice και ο Bob θέλουν να ανταλλάξουν τα κλειδιά τους χρησιμοποιώντας το ECC. Ας υποθέσουμε ότι έχουν αποφασίσει την καμπύλη E και το αρχικό σημείο P. Στη συνέχεια, η Alice επιλέγει το private key της, το a. Ομοίως, ο Μπομπ επιλέγει το private key του, το b. Και τα δύο private keys είναι δύο μεγάλοι αριθμοί (256 bit). Για τον υπολογισμό των public keys τους και οι δύο εφαρμόζουν τον πολλαπλασιασμό σημείων στο αρχικό σημείο P. Το public key της Alice, το A, υπολογίζεται από την εξίσωση A = a*P. Ομοίως, για τον υπολογισμό του public key του Bob, ο πολλαπλασιασμός σημείου εφαρμόζεται με

συντελεστή το private key του, δηλαδή B = b*P. Και τα δύο public keys, είναι σημεία στην ελλειπτική καμπύλη E.

Η Alice και ο Bob ανταλλάσσουν τα public keys τους. Και οι δύο υπολογίζουν το κοινό μυστικό τους, το W. Για τον υπολογισμό του κοινού μυστικού σημείου W, έκαναν πολλαπλασιασμό σημείου χρησιμοποιώντας τα δικά τους private keys και το public key που έλαβαν. Το κοινό μυστικό σημείο W που υπολογίζεται είναι το ίδιο κλειδί. Η Alice υπολογίζει το W = a*B και ο Bob υπολογίζει το W = b*A, όπου A = a*P και B = b*P. Έτσι, η Alice υπολογίζει W = a *(b*P) και ομοίως ο Bob υπολογίζει W = b *(a*P), έτσι και οι δύο, η Alice και ο Bob υπολογίζουν το ίδιο κοινό μυστικό σημείο που είναι το W. Οι λεπτομέρειες που είναι γνωστές είναι η καμπύλη E, το prime p, το αρχικό σημείο P και τα δύο public keys των Alice και Bob, A και B. Εάν ένας εισβολέας θέλει να θέσει σε κίνδυνο το πρωτόκολλο ECDH, πρέπει να υπολογίσει το κοινό μυστικό που υπολογίζουν η Alice και ο Bob, δηλαδή το T = a*B = b*A = a*b*P. Ο εισβολέας θα πρέπει να λύσει ένα από αυτά τα προβλήματα λογάριθμου: $a = \mathrm{Log}_p A$ ή $b = \mathrm{Log}_p B$.

Κεφάλαιο 3

Παραγωγή κλειδιών

3.1 Openssl

Το OpenSSL [5] είναι ένα ισχυρό, εμπορικής ποιότητας και πλήρες σύνολο εργαλείων για τα πρωτόκολλα Transport Layer Security (TLS) και Secure Sockets Layer (SSL). Είναι επίσης μια βιβλιοθήκη κρυπτογραφίας γενικής χρήσης. Το OpenSSL περιέχει μια εφαρμογή ανοιχτού κώδικα των πρωτοκόλλων SSL και TLS. Η βασική βιβλιοθήκη, γραμμένη στη γλώσσα προγραμματισμού C, υλοποιεί βασικές κρυπτογραφικές συναρτήσεις και παρέχει διάφορες λειτουργίες χρησιμότητας.

Η βιβλιοθήκη OpenSSL EC παρέχει υποστήριξη για την κρυπτογραφία ελλειπτικής καμπύλης (ECC). Αποτελεί τη βάση για την υλοποίηση του OpenSSL του Αλγορίθμου Ψηφιακής Υπογραφής Ελλειπτικής Καμπύλης (ECDSA) και της Ελλειπτικής Καμπύλης Diffie-Hellman (ECDH).

3.1.1 Τύποι πεδίων

Αρχικά, υπάρχουν πολλοί διαφορετικοί τύποι πεδίων που θα μπορούσαν να χρησιμοποιηθούν για τις τιμές x και y ενός σημείου (x, y). Στην πράξη, ωστόσο, χρησιμοποιούνται δύο primary, και αυτές είναι οι δύο που υποστηρίζονται από τη βιβλιοθήκη OpenSSL EC.

Το απλούστερο πεδίο αναφέρεται συνήθως ως το πρώτο (prime) πεδίο Fp όπου το p είναι ένας πρώτος (prime) αριθμός. Στις κρυπτογραφικές εφαρμογές το p πρέπει να είναι ένας πολύ μεγάλος πρώτος (prime) αριθμός. Τα στοιχεία του σετ είναι απλώς οι αριθμοί 0 έως p-1, και η προσθήκη και ο πολλαπλασιασμός πάνω στο πεδίο αυτό έχουν σημασία για την modular αριθμητική. Έτσι, εάν p = 7 τότε τα στοιχεία του συνόλου είναι {0, 1, 2, 3, 4, 5, 6} και:

0 + 1 = 1

2 + 3 = 5

3 + 3 = 6

4 + 3 = 0

5 + 4 = 2

Και ούτω κάθε εξής.

Ο επόμενος κοινός τύπος πεδίου αναφέρεται ως το δυαδικό πεδίο F_2^m. Τα στοιχεία ενός δυαδικού πεδίου αντιπροσωπεύονται συνήθως ως πολυώνυμα και όχι ως αριθμοί. Έτσι, για παράδειγμα, ένα στοιχείο θα μπορούσε να είναι:

$x^4 + x^2 + 1$

Αυτό μπορεί στη συνέχεια να εκφραστεί ως δυαδικός αριθμός ({1 0 1 0 1} σε αυτήν την περίπτωση), όπου κάθε όρος αντιπροσωπεύει ένα bit στη δυαδική αναπαράσταση. Η προσθήκη τέτοιων πολυωνύμων γίνεται κανονικά, αλλά με το αποτέλεσμα κάθε όρου μειωμένο modulo 2. Έτσι, για παράδειγμα:

$(x^2 + 1) + (x^2 + x) = 2x^2 + x + 1$

Κάθε όρος στη συνέχεια μειώνεται με modulo 2 για να δώσει μια απάντηση

$0x^2 + x + 1 = x + 1$

Σε δυαδική αναπαράσταση αυτό το άθροισμα θα μπορούσε να εκφραστεί ως εξής:

{1 0 1} + {1 1 0} = {0 1 1}

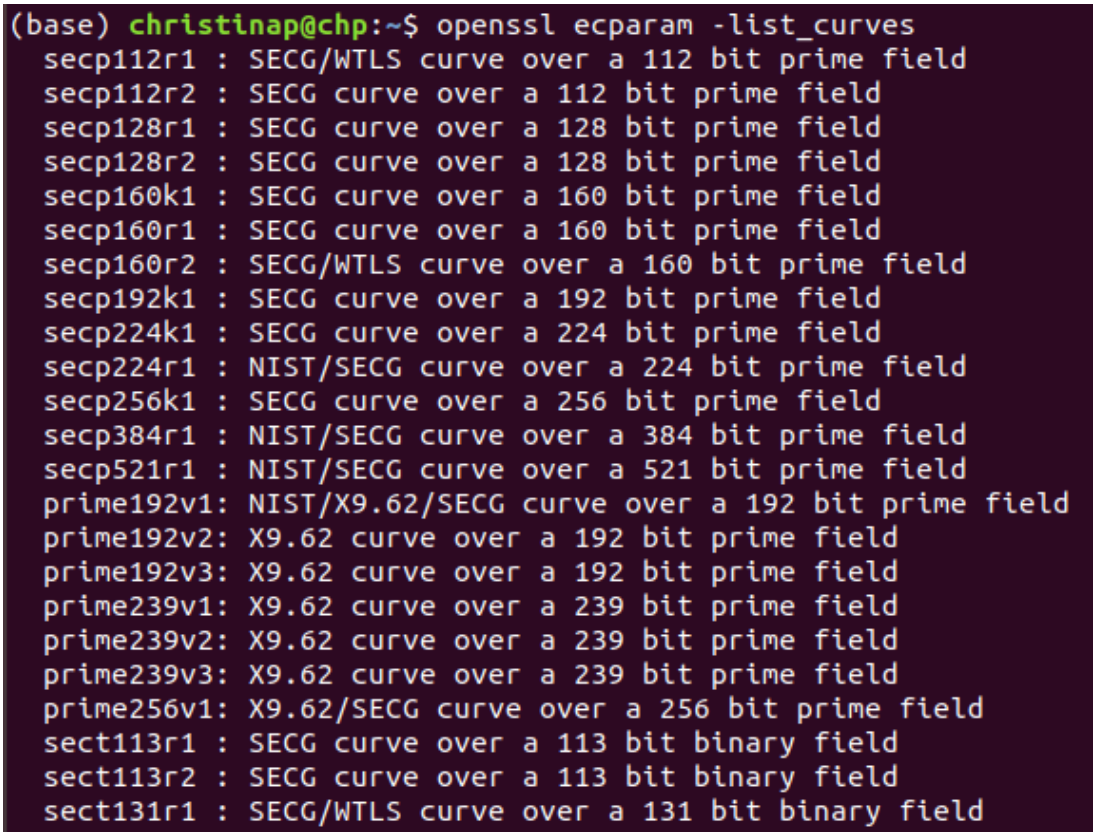
Σημειώνεται ότι η προσθήκη είναι απλώς μια απλή λειτουργία XOR.

Ο πολλαπλασιασμός στο δυαδικό πεδίο πραγματοποιείται σε σχέση με ένα αμείωτο πολυώνυμο [6]. Ο πολλαπλασιασμός των πολυωνύμων γίνεται με τον κανονικό τρόπο και στη συνέχεια το αποτέλεσμα διαιρείται με το αμείωτο πολυώνυμο. Το υπόλοιπο είναι το αποτέλεσμα του πολλαπλασιασμού.

3.1.2 Παραγωγή Elliptic Curve κλειδιών και παραμέτρων

Ένα αρχείο παραμέτρων EC περιέχει όλες τις απαραίτητες πληροφορίες για τον καθορισμό μιας ελλειπτικής καμπύλης που μπορεί στη συνέχεια να χρησιμοποιηθεί για κρυπτογραφικές λειτουργίες (για OpenSSL αυτό σημαίνει ECDH και ECDSA). Το OpenSSL περιέχει ένα μεγάλο σύνολο προκαθορισμένων καμπυλών που μπορούν να χρησιμοποιηθούν. Ο πλήρης κατάλογος των ενσωματωμένων καμπυλών μπορεί να ληφθεί με την ακόλουθη εντολή και ένα παράδειγμα αποτελέσματος της εντολής φαίνεται στην Εικόνα 3.1 :

```
$ openssl ecparam -list_curves
```



Εικόνα 3.1 Παράδειγμα εκτέλεσης εντολής `openssl ecparam -list_curves` σε Ubuntu terminal

Στη συνέχεια, μπορεί να δημιουργηθεί ένα αρχείο παραμέτρων EC για οποιαδήποτε από τις ενσωματωμένες καμπύλες με την εξής εντολή:

```
$ openssl ecparam -name secp256k1 -out secp256k1.pem
```

Τα κλειδιά μπορούν να δημιουργηθούν από την εντολή ecparam, είτε μέσω ενός προϋπάρχοντος αρχείου παραμέτρων είτε απευθείας επιλέγοντας το όνομα της καμπύλης. Για να δημιουργηθεί ένα ζεύγος private / public κλειδιού από ένα προϋπάρχον αρχείο παραμέτρων χρησιμοποιήστε η εξής εντολή:

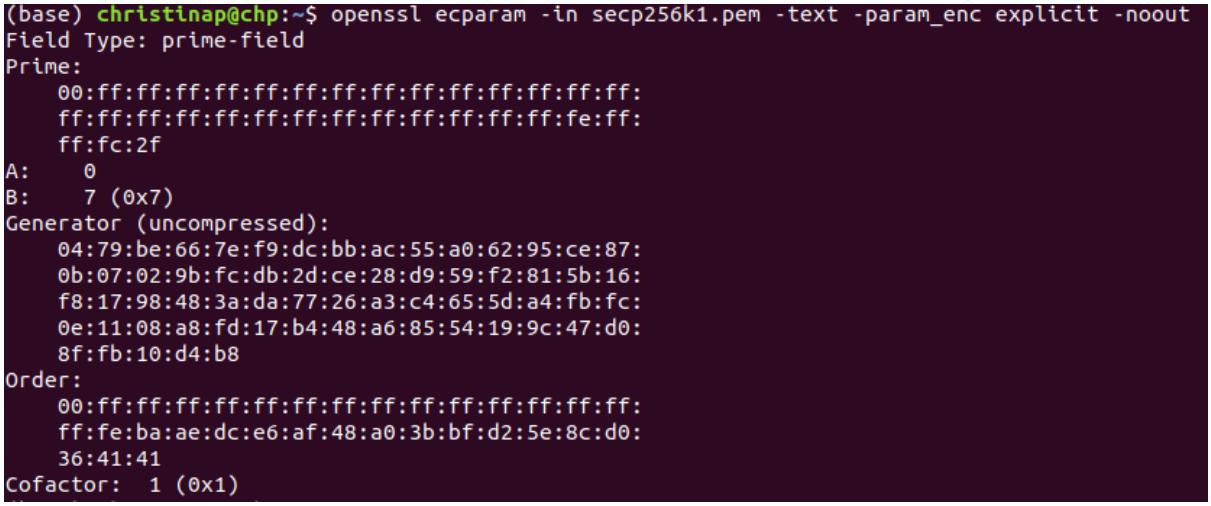
```
$ openssl ecparam -in secp256k1.pem -genkey -noout -out secp256k1-key.pem
```

Χωρίς αρχείου παραμέτρων, η εντολή είναι ως εξής:

```
$ openssl ecparam -name secp256k1 -genkey -noout -out secp256k1-key.pem
```

Για παρουσίαση συγκεκριμένων λεπτομερειών των παραμέτρων που σχετίζονται με μια συγκεκριμένη καμπύλη, μπορεί να επιτευχθεί ως εξής και ένα παράδειγμα αποτελέσματος της εντολής αυτής φαίνεται στην Εικόνα 3.2:

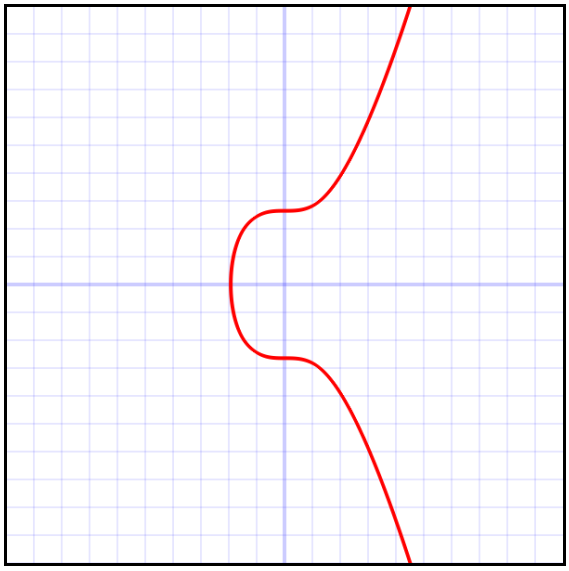
```
$ openssl ecparam -in secp256k1.pem -text -param_enc explicit -noout
```



Εικόνα 3.2 Παράδειγμα εντολής `openssl ecparam -in secp256k1.pem -text -param_enc explicit -noout` σε Ubuntu terminal

3.1.3 Επιλογή καμπύλης secp256k1

Η ελλειπτική καμπύλη που επιλέχτηκε για να γίνει η παραγωγή των κλειδιών με την βιβλιοθήκη OpenSSL, είναι η `secp256k1`. Η καμπύλη `secp256k1` αναφέρεται στις παραμέτρους της ελλειπτικής καμπύλης που χρησιμοποιείται στην κρυπτογράφηση public key του Bitcoin και ορίζεται από το Standards for Efficient Cryptography (SEC) [7]. Λόγω του γεγονότος ότι ο μηχανισμός ψηφιακής υπογραφής ECDSA έχει υιοθετηθεί από το Bitcoin, η ελλειπτική καμπύλη `secp256k1` χρησιμοποιείται ευρέως στη βιομηχανία blockchain. Οπότε φαίνεται ότι η καμπύλη `secp256k1` είναι από της πιο διαδεδομένες και για αυτό το λόγο επιλέχτηκε για σκοπούς αυτής της έρευνας.



Εικόνα 3.3 Της ελλειπτικής καμπύλης secp256k1 y2 = x3 + 7 πάνω από πραγματικούς αριθμούς.

Σημειώνεται ότι επειδή η καμπύλη secp256k1 ορίζεται στην πραγματικότητα στο πεδίο Zp, το γράφημα της θα μοιάζει με τυχαία διάσπαρτα σημεία, όχι ακριβώς όπως στην Εικόνα 3.3.

3.1.4 Αυτοματοποίηση για παραγωγή μεγάλου όγκου EC κλειδιών

Για σκοπούς αυτής της έρευνας χρειάστηκε να παραχθεί μεγάλος όγκος από EC κλειδιά τα οποία στην συνέχεια θα μπορούν να αναλυθούν και να επεξεργαστούν για ελέγχους ασφάλειας. Για την παραγωγή τους χρησιμοποιήθηκε python script, που βρίσκεται στο Παράρτημα Α, το οποίο τρέχει τις OpenSSL εντολές παραγωγής EC κλειδιών σε περιβάλλον Linux (Ubuntu v18.04). Δημιουργήθηκαν περίπου ένα εκατομμύριο OpenSSL EC κλειδιά και η αποθήκευση τους έγινε σε βάση δεδομένων με την χρήση του εργαλείου SQLite [8].

3.2 Libsodium

Το Libsodium είναι μια φορητή, installable, packable και API- compatible έκδοση του NaCl. Το NaCl ("salt" – “αλάτι”) είναι μια συντομογραφία του "Networking and Cryptography library". Είναι μια βιβλιοθήκη λογισμικού υψηλής ταχύτητας για επικοινωνία δικτύων, κρυπτογράφηση, αποκρυπτογράφηση και ψηφιακές υπογραφές [9].

Η εφαρμογή του NaCl γράφεται σε C, συχνά με αρκετούς ενσωματωμένους assemblers. Οι C++ και Python διαχειρίζονται από το NaCl ως wrappers [10]. Το NaCl μπορεί να χρησιμοποιηθεί από διάφορες γλώσσες προγραμματισμού όπως για παράδειγμα και από την PHP που αποτελεί τη βάση για το Libsodium.

3.2.1 Παραγωγή Elliptic Curve κλειδιών

Η παραγωγή ενός ζεύγους Elliptic Curve κλειδιών μπορεί να γίνει πολύ εύκολα με ένα μικρό πρόγραμμα σε Python. Η python βιβλιοθήκη που χρειάζεται είναι η pynacl [11]. Το PyNaCl είναι μια σύνδεση της Python στο libsodium, το οποίο είναι ένα fork της βιβλιοθήκης Δικτύωσης και Κρυπτογραφίας. Αυτές οι βιβλιοθήκες έχουν ως στόχο τη βελτίωση της χρηστικότητας, της ασφάλειας και της ταχύτητας των συστημάτων και των δικτύων. Υποστηρίζει Python 2.7 και 3.5+ καθώς και PyPy 2.6+. Παράδειγμα ενός Python προγράμματος για παραγωγή EC κλειδιών φαίνεται στην Εικόνα 3.4 και αποτέλεσμα του προγράμματος στην Εικόνα 3.5.

```
1  from nacl.public import PrivateKey
2  import binascii
3
4  privKey = PrivateKey.generate()
5  pubKey = privKey.public_key
6
7  print("privKey:", binascii.hexlify(bytes(privKey)))
8  print("pubKey: ", binascii.hexlify(bytes(pubKey)))
```

Εικόνα 3.4 Παράδειγμα Python προγράμματος για παραγωγή EC κλειδιών με NaCl

```
privKey: b'8175f7cd524a59b6efbd447985ce5d97c546b319521ff236203970e50052c641'
pubKey:  b'cf97a96568fee4ddb232f617fd5b9df2d2e5b90e68ba7f6d5129ea92d7d8f95e'
```

Εικόνα 3.5 Παράδειγμα αποτελέσματος προγράμματος Εικόνα 3.4

3.2.2 Επιλογή καμπύλης Curve25519

Το Libsodium περιλαμβάνει ένα εξειδικευμένο interface για το Elliptic Curve Diffie-Hellman για την καμπύλη Curve25519 [12] που χρησιμοποιείται στην εφαρμογή κρυπτογράφησης public key και σε πρωτόκολλα ανταλλαγής κλειδιών. Η ελλειπτική καμπύλη Curve25519 είναι μια ελλειπτική καμπύλη που προσφέρει 128 bits ασφάλειας και έχει σχεδιαστεί να

χρησιμοποιείται με την ελλειπτική καμπύλη Diffie – Hellman (ECDH). Παράλληλα η Curve25519 είναι μια από τις ταχύτερες καμπύλες ECC [13].

Το Libsodium δεν προσφέρει υποστήριξη καμπύλης στα επίπεδα ασφαλείας 224-bit και 256-bit άρα δεν υποστηρίζει την καμπύλη secp256k1 που χρησιμοποιήθηκε στην OpenSSL στο κεφάλαιο 3.1, οπότε για τους σκοπούς της έρευνας αυτής έχει επιλεχθεί η καμπύλη Curve25519.

3.2.3 Αυτοματοποίηση για παραγωγή μεγάλου όγκου EC κλειδιών

Επίσης για αυτή την μελέτη χρειάστηκε να παραχθεί μεγάλος όγκος από EC κλειδιά τα οποία στην συνέχεια θα μπορούν να αναλυθούν και να επεξεργαστούν για ελέγχους ασφάλειας. Για την παραγωγή τους χρησιμοποιήθηκε python script, που βρίσκεται στο Παράρτημα Β, το οποίο δημιουργεί περίπου ένα εκατομμύριο Libsodium / NaCl EC κλειδιά και η αποθήκευση τους έγινε σε βάση δεδομένων με την χρήση του εργαλείου SQLite [8].

3.3 BouncyCastle

Το Bouncy Castle Crypto package είναι μια εφαρμογή κρυπτογραφικών αλγορίθμων της Java [14]. Η μαθηματική υποστήριξη για ελλειπτικές καμπύλες παρέχεται στο package org.bouncycastle.math.ec [15] στο οποίο υπάρχουν οι κλάσεις και τα interfaces που σχετίζονται με την κρυπτογραφία στα πακέτα org.bouncycastle.jce.spec και org.bouncycastle.jce.interfaces.

3.3.1 Παραγωγή Elliptic Curve κλειδιών

Interfaces:

- ECKey Interface

Το Interface ECKey έχει μία μόνο μέθοδο, την getParameters(), η οποία επιστρέφει ένα ECParameterSpec που αντιπροσωπεύει τις παραμέτρους τομέα για την ελλειπτική καμπύλη με την οποία συνδέεται το κλειδί.

- ECPrivateKey Interface

Το Interface ECPrivateKey έχει μία μόνο μέθοδο, την getD(), η οποία επιστρέφει ένα BigInteger που αντιπροσωπεύει την private τιμή για το private key ελλειπτικής καμπύλης.

- ECPublicKey Interface

Το Interface ECPublicKey έχει μία μόνο μέθοδο, την getQ(), η οποία επιστρέφει το ECPoint που αντιπροσωπεύει το public σημείο για το public key ελλειπτικής καμπύλης.

- ECPointEncoder Interface

Το Interface ECPointEncoder έχει μια μέθοδο, την setPointFormat(), η οποία παίρνει μια συμβολοσειρά που αντιπροσωπεύει το στυλ κωδικοποίησης που μπορεί να χρησιμοποιηθεί όταν το κλειδί EC που εφαρμόζει αυτήν το Interface έχει τη μέθοδο getEncoded() [16].

3.3.1.1 Παραγωγή ζεύγους κλειδιών

Η δημιουργία ζεύγους κλειδιών μπορεί να γίνει χρησιμοποιώντας παραμέτρους που έχουν δημιουργηθεί ρητά ή ανακτώντας μια καμπύλη που υπάρχει όπως την ελλειπτική καμπύλη secp256k1.

Από ρητές παραμέτρους

Απαιτείται το org.bouncycastle.jce.ECParameterSpec για την κατασκευή κλειδιού ελλειπτικής καμπύλης. Ο πιο χρονοβόρος τρόπος δημιουργίας είναι να δημιουργήθεί το αντικείμενο ECParameterSpec από ένα αντικείμενο Bouncy Castle ECCurve και ένα σχετικό σημείο βάσης.

Κανονικά αυτό γίνεται μόνο εάν η καμπύλη που θα ήθελε κάποιος δεν υπάρχει ήδη σε έναν από τους πίνακες καμπυλών που υπάρχουν, αλλά αν ήδη υπάρχει ένα σύνολο παραμέτρων το παράδειγμα παραγωγής ζεύγους κλειδιού φαίνεται στο Εικόνα 3.6.

```
import org.bouncycastle.math.ec.ECCurve;
import org.bouncycastle.jce.spec.ECParameterSpec;
...
ECCurve curve = new ECCurve.Fp(
            new BigInteger("883423532389192164791648750360308885314476597252960362792450860609699839"), // q
            new BigInteger("7fffffffffffffffffffffff7fffffffffff8000000000007ffffffffc", 16), // a
            new BigInteger("6b016c3bdcf18941d0d654921475ca71a9db2fb27d1d37796185c2942c0a", 16)); // b
ECParameterSpec ecSpec = new ECParameterSpec(
            curve,
            curve.decodePoint(Hex.decode("020ffa963cdca8816ccc33b8642bedf905c3d358573d3f27fbbd3b3cb9aaaf")), // G
            new BigInteger("883423532389192164791648750360308884807550341691627752275345424702807307")); // n
KeyPairGenerator g = KeyPairGenerator.getInstance("ECDSA", "BC");
g.initialize(ecSpec, new SecureRandom());
KeyPair pair = g.generateKeyPair();
```

Εικόνα 3.6 Παράδειγμα BouncyCastle παραγωγής ζεύγους κλειδιού με παραμέτρους

Όπως φαίνεται στην Εικόνα 3.6, είναι μια διαδικασία δύο βημάτων. Πρώτα πρέπει να δημιουργηθεί η καμπύλη και έπειτα πρέπει να συσχετιστεί η καμπύλη με ένα σημείο βάσης χρησιμοποιώντας ένα ECParameterSpec το οποίο στη συνέχεια χρησιμοποιείται για την προετοιμασία του αντικειμένου KeyPairGenerator.

Από καμπύλη

Οι καμπύλες με ονομασία χειρίζονται από τον provider Bouncy Castle συσχετίζοντας ένα σύνολο παραμέτρων με ένα όνομα χρησιμοποιώντας μια επέκταση του ECParameterSpec, το ECNamedCurveParameterSpec, το οποίο μπορεί να βρεθεί στο org.bouncycastle.jce.spec.

Παράδειγμα παραγωγής ζεύγους κλειδιού με την χρήση της καμπύλης secp256k1 φαίνεται στο Εικόνα 3.7.

```
import org.bouncycastle.jce.spec.ECParameterSpec;
...
ECParameterSpec ecSpec = ECNamedCurveTable.getParameterSpec(" secp256k1 ");
KeyPairGenerator g = KeyPairGenerator.getInstance("ECDSA", "BC");
g.initialize(ecSpec, new SecureRandom());
KeyPair pair = g.generateKeyPair();
```

Εικόνα 3.7 παράδειγμα BouncyCastle παραγωγής ζεύγους κλειδιού με ονομαστική καμπύλη

3.3.2 Επιλογή καμπύλης secp256k1

Η ελλειπτική καμπύλη που επιλέχτηκε για να γίνει η παραγωγή των κλειδιών με την βιβλιοθήκη BouncyCastle, είναι η secp256k1. Οι λόγοι για την επιλογή αυτής της συγκεκριμένης ελλειπτικής καμπύλης είναι αυτοί που έχουν προαναφερθεί στο 3.1.

3.3.3 Αυτοματοποίηση για παραγωγή μεγάλου όγκου EC κλειδιών

Για αυτή την έρευνα χρειάστηκε να παραχθεί μεγάλος όγκος από EC κλειδιά τα οποία στην συνέχεια θα μπορούν να αναλυθούν και να επεξεργαστούν για ελέγχους ασφάλειας. Για την

παραγωγή τους χρησιμοποιήθηκε Java κώδικας BouncyCastle που βρίσκεται στο Παράρτημα Γ, ο οποίος δημιουργεί περίπου ένα εκατομμύριο κλειδιά BouncyCastle και η αποθήκευση τους έγινε σε βάση δεδομένων με την χρήση του εργαλείου SQLite [8].

Κεφάλαιο 4

White box

Το White box testing μια μέθοδος που ελέγχει είναι ένα υποσύστημα του οποίου τα εσωτερικά μπορούν να προβληθούν αλλά συνήθως δεν μπορούν να τροποποιηθούν. Έχοντας πρόσβαση στα εσωτερικά ενός υποσυστήματος γενικά καθιστά το υποσύστημα ευκολότερο να κατανοηθεί, αλλά και ευκολότερο να διεισδύσει κάποιος μέσα. Για παράδειγμα, εάν ένας προγραμματιστής μπορεί να εξετάσει τον πηγαίο κώδικα, οι αδυναμίες σε έναν αλγόριθμο είναι πολύ πιο εύκολο να ανακαλυφθούν. Αυτό καθιστά το White box testing πολύ πιο αποτελεσματικό από το Black Box testing αλλά πολύ πιο δύσκολο λόγο της πολυπλοκότητας που απαιτείται από την πλευρά του ελεγκτή για να κατανοήσει το υποσύστημα [17].

Η δοκιμή White box είναι μία από τις δύο μεγαλύτερες μεθόδους δοκιμών που χρησιμοποιούνται σήμερα. Έχει πολλά σημαντικά πλεονεκτήματα όπως:

- Οι παρενέργειες της γνώσης του πηγαίου κώδικα είναι επωφελείς για ενδελεχή έλεγχο ενός συστήματος.
- Η βελτιστοποίηση του κώδικα γίνεται εύκολη καθώς εκδηλώνονται δυσδιάκριτα σημεία συμφόρησης.
- Εύκολο στην αυτοματοποίηση [18].
- Παρέχει σαφείς, βασισμένους σε engineering-based κανόνες για το πότε θα σταματήσει η δοκιμή.

Αν και η δοκιμή White box έχει σημαντικά πλεονεκτήματα, δεν είναι τέλεια και περιέχει ορισμένα μειονεκτήματα όπως:

- Η δοκιμή White box φέρνει πολυπλοκότητα στις δοκιμές, επειδή ο υπεύθυνος δοκιμών πρέπει να έχει γνώση του προγράμματος ή η ομάδα δοκιμής πρέπει να έχει τουλάχιστον έναν πολύ καλό προγραμματιστή που μπορεί να κατανοήσει το πρόγραμμα σε επίπεδο κώδικα. Η δοκιμή White box απαιτεί έναν προγραμματιστή με υψηλό επίπεδο γνώσεων και χρόνια εμπειρίας λόγω της πολυπλοκότητας του επιπέδου δοκιμών που πρέπει να γίνουν.
- Σε ορισμένες περιπτώσεις, δεν είναι δυνατό να μπορεί να δοκιμαστεί κάθε υπάρχουσα συνθήκη της εφαρμογής οπότε ορισμένες περιπτώσεις μπορεί να μην ελεγχθούν σωστά θα δοκιμαστούν.
- Οι δοκιμές επικεντρώνονται στο λογισμικό καθώς υπάρχει και ενδέχεται να μην εντοπιστούν λειτουργίες που λείπουν.
- Η δοκιμή μπορεί να είναι εύθραυστη επειδή συνδέεται στενά με τη συγκεκριμένη εφαρμογή του αντικειμένου που δοκιμάζεται.

Εικόνα 4.1 Γράφημα για μορφή του White Box

Η White Box προσέγγιση που έχει ακολουθηθεί για σκοπούς αυτής της έρευνας είναι η ανάλυση του Random number generator για την κάθε βιβλιοθήκη που χρησιμοποιείται. Βλέποντας την μέθοδο που χρησιμοποιεί κάθε βιβλιοθήκη για την παραγωγή των κλειδιών και συγκεκριμένα των EC κλειδιών, θα μπορούν να εντοπιστούν τυχών σφάλματα ή αδυναμίες που θα οδηγήσουν στο αν ο αλγόριθμος μπορεί να είναι ευάλωτος σε επιθέσεις.

4.1 Random number generator Opensll

Το OpenSSL [5] είναι ένα ισχυρό, εμπορικής ποιότητας και πλήρες σύνολο εργαλείων για τα πρωτόκολλα Transport Layer Security (TLS) και Secure Sockets Layer (SSL). Είναι επίσης μια βιβλιοθήκη κρυπτογραφίας γενικής χρήσης. Το OpenSSL περιέχει μια εφαρμογή ανοιχτού κώδικα των πρωτοκόλλων SSL και TLS. Η βασική βιβλιοθήκη, γραμμένη στη γλώσσα προγραμματισμού C, υλοποιεί βασικές κρυπτογραφικές συναρτήσεις και παρέχει διάφορες λειτουργίες χρησιμότητας.
Το OpenSSL χρησιμοποιεί τη γεννήτρια md_rand. Το md_rand χρησιμοποιεί τον MD5 κατακερματισμό ως ψευδοτυχαία συνάρτηση. Ο πηγαίος κώδικας βρίσκεται στο crypto/rand/md_rand.c.

Η RAND_METHOD του OpenSSL, όπως ορίζεται από το RAND_set_rand_method() και επιστρέφεται από το RAND_get_rand_method(), χρησιμοποιείται μόνο εάν κανένα ENGINE δεν έχει οριστεί ως η προεπιλεγμένη εφαρμογή "rand".

Οι μηχανισμοί που περιγράφονται παρακάτω αφορούν αποκλειστικά την εφαρμογή PRNG ενσωματωμένη στο OpenSSL και είναι υπεύθυνοι για τη δημιουργία και τη διαχείριση ενός κρυπτογραφικά ασφαλούς PRNG stream.

Αυτές οι λειτουργίες εφαρμόζουν μια κρυπτογραφικά ασφαλή γεννήτρια ψευδο-τυχαίων αριθμών (PRNG). Χρησιμοποιείται από άλλες λειτουργίες βιβλιοθήκης για παράδειγμα για τη δημιουργία τυχαίων κλειδιών EC και οι εφαρμογές μπορούν να το χρησιμοποιήσουν όταν χρειάζονται τυχαιότητα. Ένα κρυπτογραφικό PRNG πρέπει να δημιουργηθεί με απρόβλεπτα δεδομένα, όπως κινήσεις ποντικιού ή πλήκτρα που πιέζονται τυχαία από τον χρήστη.

Η εφαρμογή του RNG βρίσκεται στο αρχείο md rand.c. Ορίζει το προεπιλεγμένο RNG που θα χρησιμοποιηθεί στο OpenSSL. Όπως κάθε καθαρά RNG που βασίζεται σε λογισμικό βασίζεται σε μια γεννήτρια ψευδοτυχαίων αριθμών (PRNG). Το API των συναρτήσεων που

σχετίζονται με το RNG του OpenSSL περιγράφονται στην αντίστοιχη σελίδα [19]. Οι συναρτήσεις περιγράφονται παρακάτω:

- void RAND_add(const void *buf, int num, double entropy)

«Προσθέτει» την εντροπία που περιέχεται στο buf of length num στην εσωτερική κατάσταση του RNG, όπου η εντροπία θα είναι μια εκτίμηση της πραγματικής εντροπίας που περιέχεται στο buf. Η εντροπία είναι το μέτρο της «τυχαιότητας» σε μια ακολουθία bits. Τα δεδομένα που τροφοδοτήθηκαν πρόσφατα τροποποιούν την κατάσταση του RNG έτσι ώστε οποιαδήποτε τυχαία έξοδος που δημιουργείται στη συνέχεια να επηρεάζεται από αυτήν. Σε αυτό το έργο δείχνουμε ότι η «προσθήκη» της εντροπίας υποφέρει από σοβαρή αδυναμία.

- void RAND_seed(const void *buf, int num)

είναι ένα wrapper για την RAND add. Καλεί αυτή τη λειτουργία με εντροπία = num, δηλαδή αναμένει ότι τα seed data θα έχουν μέγιστη εντροπία.

- int RAND_poll()

αντλεί εντροπία από τις πηγές τυχαιότητας του λειτουργικού συστήματος, π.χ. /dev/random σε Linux και την τροφοδοτεί στο RNG.

- int RAND_bytes(unsigned char *buf, int num)

εξάγει num τυχαία byte στο buf. Εάν το επίπεδο εντροπίας του RNG, υπολογιζόμενο ως το άθροισμα των εκτιμήσεων που παρέχονται από τα calls προς RAND add, είναι μικρότερο από το καθορισμένο ελάχιστο των 256 bit, αυτή η συνάρτηση επιστρέφει με κωδικό σφάλματος.

- int RAND_pseudo_bytes(unsigned char *buf, int num)

εκτελεί την ίδια λειτουργία για την παραγωγή τυχαίας εξόδου με τα RAND bytes, εκτός από το ότι παράγει επίσης έξοδο εάν δεν έχει επιτευχθεί το ελάχιστο επίπεδο εντροπίας.

4.1.1 Περιορισμός της εξόδου του RNG σε Εντροπία 240 Bits

Το πρώτο ελάττωμα σχεδίασης του πυρήνα RNG του OpenSSL οδηγεί στην πιθανότητα τα κλειδιά που δημιουργούνται να περιορίζονται σε 240 bits ασφαλείας, ενώ προσπαθεί να επιτύχει 256 bit - ωστόσο, λόγω έλλειψης documentation, το επίπεδο ασφάλειας μπορεί να συναχθεί μόνο από τον πηγαίο κώδικα. Αυτό το ζήτημα προκύπτει ακόμη και όταν το RNG

αρχικά τροφοδοτείται με σωστά εκτιμώμενα δεδομένα υψηλής εντροπίας πριν από οποιαδήποτε παραγωγή. Από πρακτική άποψη, αυτό το πρόβλημα δεν έχει νόημα, καθώς δεν επιτρέπει πρακτικές επιθέσεις, ούτε καν στο προβλέψιμο μακρινό μέλλον, αλλά μας δείχνει ένα σχεδιαστικό ελάττωμα του RNG. Από αυστηρά τυπική άποψη, διαπιστώνουμε ότι είναι κοινώς αποδεκτό ότι τα κλειδιά με ασφάλεια 256 bit θα χρησιμοποιούνται για εφαρμογές όπου η μακροπρόθεσμη ασφάλεια έχει σημασία, καθώς αντικατοπτρίζεται από τα τυποποιημένα μεγέθη κλειδιών Elliptic Curve, επομένως, ένα RNG θα πρέπει να παράγει έξοδο με το αντίστοιχο επίπεδο εντροπίας. Έτσι, καταλήγουμε στο συμπέρασμα ότι αυτή η αδυναμία μπορεί να θεωρηθεί ότι οφείλεται σε σφάλμα σχεδιασμού που προκαλεί την ασφάλεια της εξόδου RNG να εξαρτάται από την ακολουθία κλήσεων. Στην πραγματικότητα, είναι πολύ πιθανό μια εφαρμογή πελάτη να παράγει πολλαπλά συμμετρικά κλειδιά για διαφορετικούς λόγους. Στη θεωρητική άποψη των RNG, λεπτομέρειες εφαρμογής όπως συγκεκριμένες ακολουθίες κλήσεων για την σχεδίαση τυχαίων bytes δεν έχουν σημασία.

4.1.2 Αποτυχία ανάκτησης χαμηλής εντροπίας

Παρακάτω, διερευνάται τι συμβαίνει όταν το RNG του OpenSSL εκτελεί τη διαδικασία ανάδευσης σε κατάσταση χαμηλής εντροπίας και στη συνέχεια η εφαρμογή πελάτη προσθέτει seed data υψηλής εντροπίας στο RNG πριν από τη δημιουργία κρυπτογραφικών κλειδιών 256 bit.

Μια βασικά χρήσιμη έννοια για μια συνάρτηση τροφοδοσίας εντροπίας σε ένα RNG είναι αυτή μιας συνάρτησης ανάμιξης, που ορίζεται ως εξής [20]:
Μια συνάρτηση ανάμειξης f εγγυάται ότι
H(f(I, S)) > H(S) και H(f(I, S) > H(I),
όπου H() δηλώνει την εντροπία Shannon, I τα seed data εισόδου και S την κατάσταση του RNG. Μια συνάρτηση που ακολουθεί αυτήν την έννοια εγγυάται ότι δεν μειώνει την εντροπία της κατάστασης RNG και ότι μετά τη λειτουργία της η κατάσταση θα έχει τουλάχιστον την ίδια εντροπία με τα seed data εισόδου.

Από καθαρά τυπική άποψη, το RNG του OpenSSL πληροί και τις δύο προϋποθέσεις. Ωστόσο, ο ορισμός της λειτουργίας ανάμειξης όπως παρέχεται στην αναφορά και φαίνεται παραπάνω αποδεικνύεται περιορισμένης χρησιμότητας: Δεδομένου ενός RNG που χρησιμοποιεί μόνο ένα μέρος της εσωτερικής του κατάστασης για την παραγωγή εξόδου, όπως συμβαίνει με το RNG του OpenSSL, ο ορισμός θα πρέπει να αναφέρεται στην εντροπία

της παραγωγής που παράγεται πρόσφατα αντί εκείνης της κατάστασης S. Σε σχέση με αυτόν τον προσαρμοσμένο ορισμό μιας συνάρτησης ανάμιξης, το RAND_add πληροί την πρώτη απαίτηση, αλλά όχι τη δεύτερη: όταν το RNG είναι μια κατάσταση χαμηλής εντροπίας, ακόμα και μετά την προσθήκη δεδομένων υψηλής εντροπίας, το RNG θα παραμείνει ουσιαστικά σε κατάσταση χαμηλής εντροπίας βραχυπρόθεσμα, δηλαδή θα παράγει χαμηλής εντροπίας έξοδο [21].

4.2 Random number generator Libsodium

Το Libsodium είναι ένα ανοιχτού κώδικα, εύκολο στη χρήση fork της βιβλιοθήκης κρυπτογράφησης NaCl και παρέχει πολλές κρυπτογραφικές λειτουργίες. Αυτές περιλαμβάνουν κρυπτογράφηση public key, digital signing, παραγωγή κλειδιού, κατακερματισμός κωδικού πρόσβασης, κωδικούς ελέγχου ταυτότητας μηνυμάτων, κρυπτογράφηση ροής, ψευδο-τυχαίες γεννήτριες αριθμών, δημιουργία τυχαίων αριθμών και υποστήριξη για Elliptic Curves όπως το Curve25519. Ο έλεγχος ασφαλείας περιλαμβάνει τα στοιχεία του libsodium που χειρίζονται επικυρωμένη συμμετρική κρυπτογράφηση, κρυπτογράφηση public key, κατακερματισμό, παραγωγή κλειδιού και ανταλλαγή κλειδιών. Γενικά, το libsodium είναι μια ασφαλής, υψηλής ποιότητας βιβλιοθήκη που ικανοποιεί τους δηλωμένους στόχους χρηστικότητας και αποτελεσματικότητας και δεν φαίνεται να υπάρχουν σοβαρές ευπάθειες. Έχουν εντοπιστεί μερικά προβλήματα χαμηλής σοβαρότητας στο libsodium που σχετίζονται με τη χρηστικότητα του API. Ωστόσο, υπάρχουν ορισμένοι πιθανοί κίνδυνοι όσο αφορά τη δημιουργία τυχαίων αριθμών που πρέπει να λάβουν υπόψη οι προγραμματιστές για να αποτρέψουν την εισαγωγή ευπαθειών στην ασφάλεια που παρέχεται.

4.2.1 Elliptic Curve Diffie-Hellman

Το Libsodium περιλαμβάνει ένα εξειδικευμένο interface για το Elliptic Curve Diffie-Hellman για το Curve25519 [22] που χρησιμοποιείται στην εφαρμογή κρυπτογράφησης public key και σε πρωτόκολλα ανταλλαγής κλειδιών. Μια σημαντική πτυχή της λειτουργικής κλίμακας πολλαπλασιασμού είναι ότι βελτιστοποιείται για διαφορετικές αρχιτεκτονικές και επεξεργαστές. Όσον αφορά την ασφάλεια, μία διαφορά στο Libsodium σε σύγκριση με το NaCl είναι ότι το υπολογισμένο κοινό μυστικό απορρίπτεται εάν είναι ίσο με 0 (διασφαλίζοντας έτσι ότι τα public key Curve25519 είναι έγκυρα). Αυτή η επικύρωση καταργεί την πιθανότητα ενός εισβολέα να ελέγχει το κλειδί χωρίς ανίχνευση από τους επικοινωνούντες peers.

4.2.2 Πιθανοί κίνδυνοι με το custom RNG API

Το interface για την προσαρμογή της γεννήτριας τυχαίων αριθμών παρέχει ένα μέσο για αλλαγή στον τρόπο με τον οποίο το Libsodium αποκτά τυχαιότητα για μια δεδομένη πλατφόρμα. Για ορισμένες ενσωματωμένες πλατφόρμες, η χρήση αυτού του API μπορεί να είναι η μόνη επιλογή για ασφαλή τυχαιότητα. Το Libsodium αποστέλλει μια ασφαλή κατασκευή ψευδοτυχαίας γεννήτριας αριθμού (PRNG) που λαμβάνει μια μικρή ποσότητα εντροπίας από το /dev/random ή το /dev/urandom (εάν υπάρχει) για να κάνει seed το Salsa20 stream cipher. Επιπλέον, η προεπιλεγμένη εφαρμογή περιλαμβάνει το random_stir() για περιστασιακή επανεκκίνηση του PRNG. Για να χρησιμοποιήσει αυτό το PRNG, ο χρήστης/πλατφόρμα απλώς καλεί το
randombytes_set_implementation(&randombytes_salsa20_implementation)
πριν από την αρχικοποίηση του πυρήνα του Libsodium μέσω του sodium_init(). Η υλοποίηση του RNG αποτελείται από 6 function pointers και απαιτούνται μόνο 3.

```
typedef struct randombytes_implementation {
        const char *(*implementation_name)(void);
        uint32_t (*random)(void);
        void (*stir)(void);
        uint32_t (*uniform)(const uint32_t upper_bound);
        void (*buf)(void * const buf, const size_t size);
        int (*close)(void);
} randombytes_implementation;
```

Εικόνα 4.2 Υλοποίηση του randombytes_set_implementation()

Σύμφωνα με έρευνες που έχουν γίνει υπάρχουν γνωστά ζητήματα με αυτήν την προσέγγιση. Δεν είναι thread-safe και η εφαρμογή randombytes_stir() είναι προαιρετική. Σε γενικές γραμμές, μπορεί να γίνει κατάχρηση αυτού του API με τρόπους που θα οδηγούσαν σε αδύναμο PRNG για ευαίσθητες κρυπτογραφικές λειτουργίες [23].

4.2.3 Έλλειψη ελλειπτικών καμπυλών σε υψηλότερα επίπεδα ασφάλειας

Η κύρια καμπύλη που υποστηρίζεται από τη βιβλιοθήκη είναι η Curve25519, η οποία είναι ισοδύναμη με το επίπεδο ασφαλείας 128-bit. Προς το παρόν, το Libsodium δεν προσφέρει υποστήριξη καμπύλης στα επίπεδα ασφαλείας 224-bit και 256-bit. Από θέμα ασφάλειας θα ήταν πιο ασφαλές να υποστήριζε μία επιπλέον καμπύλη και στο επίπεδο ασφαλείας των 256-bit. Αυτό θα παρέχει επαρκείς επιλογές καμπύλης σε διαφορετικά επίπεδα ασφάλειας για προγραμματιστές εφαρμογών και συστημάτων.

4.2.4 Συμπέρασμα για Libsodium

Από κριτικές και αναλύσεις στον κώδικα της Libsodium δεν έχουν αποκαλυφθεί κρίσιμα ελαττώματα ή ευπάθειες στη βιβλιοθήκη. Οι προγραμματιστές του Libsodium φαίνεται να έχουν ακολουθήσει τις βέλτιστες πρακτικές όσον αφορά τον σχεδιασμό κρυπτογράφησης και την ασφαλή ανάπτυξη [24].

4.3 Random number generator BouncyCastle

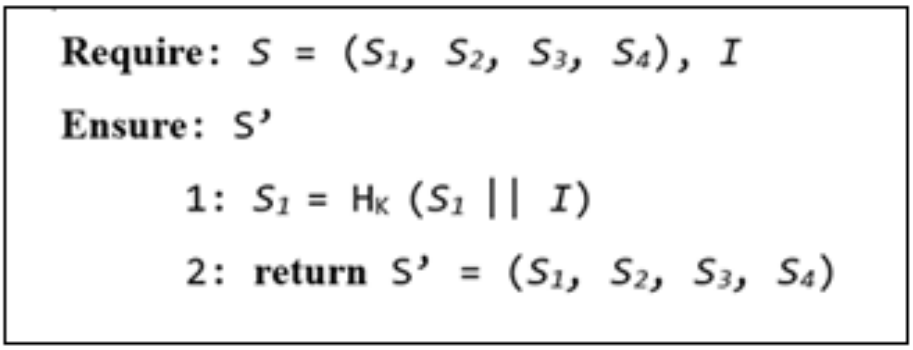
Το Bouncy Castle Crypto package είναι μια εφαρμογή κρυπτογραφικών αλγορίθμων της Java [25]. Η υλοποίηση πολλών γεννητριών ψευδο-τυχαίων αριθμών με είσοδο παρέχεται στο πακέτο org.bouncycastle.crypto.prng [26], όπου η υλοποίηση της ψευδο-τυχαίας γεννήτριας αριθμού με είσοδο SHA1PRNG είναι στην κλάση DigestRandomGenerator. Η υλοποίηση συνδυάζει μια συνάρτηση κρυπτογραφικού κατακερματισμού (η οποία είναι από προεπιλογή H_k) με εσωτερικές οδηγίες που χρησιμοποιούνται για την ενημέρωση της εσωτερικής κατάστασης της γεννήτριας. Σε σχετική έρευνα [27] που έχει γίνει στην ανάλυση του πηγαίου κώδικα, εντοπίστηκε μια αδυναμία που σχετίζεται με την αποσύνθεση της εσωτερικής κατάστασης και μια αδυναμία λόγω μιας ατελούς ενημέρωσης κατάστασης κατά τη διάρκεια του αλγορίθμου *refresh*. Αυτές οι αδυναμίες όπως αναφέρεται δεν είχαν εντοπιστεί από την κοινότητα του Bouncycastle.

4.3.1 Αποσύνθεση εσωτερικής κατάστασης

Η εσωτερική κατάσταση της γεννήτριας υλοποιείται με τα ακόλουθα πεδία: seed μεγέθους 160 bits, κατάσταση μεγέθους 160 bits, seed μετρητή μεγέθους 64 bits και μετρητής κατάστασης πεδίου, μεγέθους 64 bits. Τα δύο πρώτα πεδία περιέχουν τη συγκεντρωμένη εντροπία και τα δύο τελευταία είναι μετρητές που χρησιμοποιούνται για τη λειτουργία του. Το συνολικό μέγεθος της εσωτερικής κατάστασης είναι 448 bits και η αποσύνθεσή του είναι $S = (S_1, S_2, S_3, S_4)$, όπου S_1, S_2, S_3, S_4 αντιπροσωπεύουν Seed, κατάσταση, seedCounter και stateCounter, αντίστοιχα.

4.3.2 Αλγόριθμος refresh

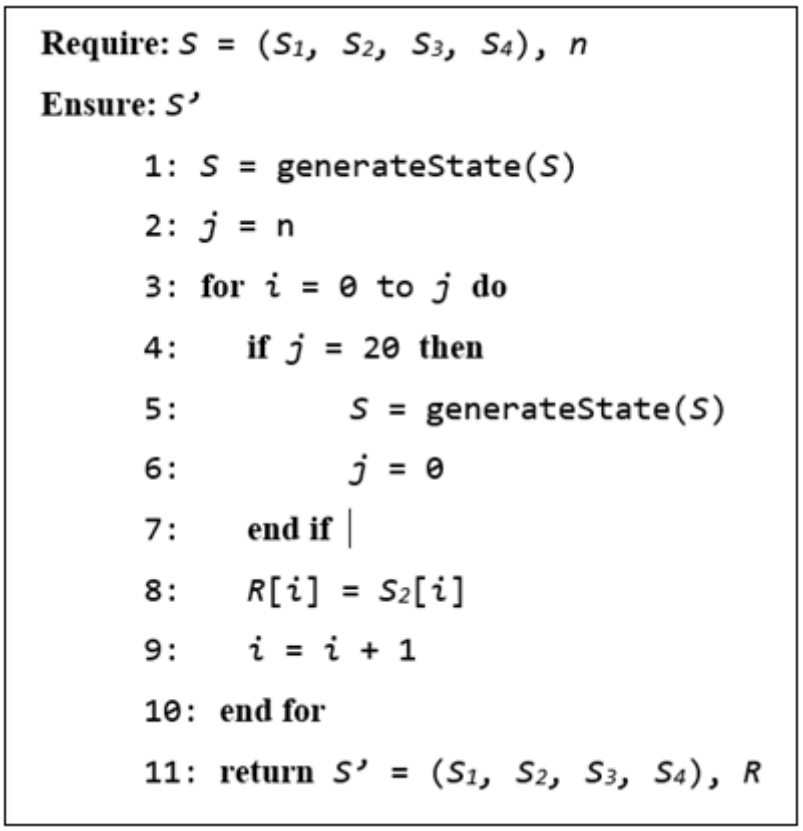
Αυτός ο αλγόριθμος περιγράφεται πλήρως στην Εικόνα 4.3. Παίρνει ως είσοδο την τρέχουσα εσωτερική κατάσταση (S_1, S_2, S_3, S_4) και μια είσοδο I. Εξάγει μια νέα εσωτερική κατάσταση όπου ενημερώνεται μόνο το S1. Υλοποιείται με τη μέθοδο addSeedMaterial.



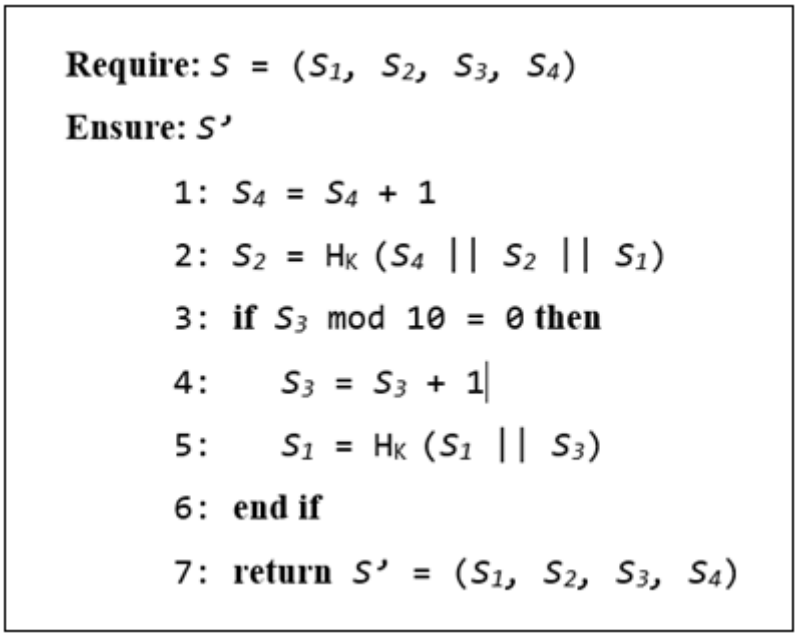
Εικόνα 4.3 Bouncycastle SHA1PRNG refresh

4.3.3 Αλγόριθμος next

Αυτός ο αλγόριθμος περιγράφεται στην Εικόνα 4.4 και Εικόνα 4.5. Υλοποιείται με τη μέθοδο NextBytes. Παίρνει ως είσοδο έναν ακέραιο n, την τρέχουσα εσωτερική κατάσταση (S_1, S_2, S_3, S_4) και εξάγει μια συμβολοσειρά n-byte R. Η έξοδος R προέρχεται από το S_2, ενώ μια εσωτερική μέθοδος, που ονομάζεται createState χρησιμοποιείται για την ενημέρωση του state.



Εικόνα 4.4 Bouncycastle SHA1PRNG next (NextBytes)[2]



Εικόνα 4.5 Bouncycastle SHA1PRNG next: (generateState)

Η εντολή createState αυξάνει τους μετρητές S_3 και S_4 και υπολογίζει τις νέες τιμές των S_1 και S_2 αναλόγως.

4.3.4 Επιθέσεις στο Bouncycastle

Έχουν γίνει επιθέσεις εναντίον του Bouncycastle SHA1PRNG λαμβάνοντας υπόψη την αποσύνθεση της εσωτερικής κατάστασης [32]. Ο εισβολέας χρησιμοποιεί μια έξοδο που δημιουργήθηκε προηγουμένως ως είσοδο για να καταστρέψει τη γεννήτρια: η επίθεσή παρουσιάζει ότι το Bouncycastle SHA1PRNG δεν είναι ανθεκτικό [33]

Κεφάλαιο 5

Black box

Η δοκιμή Black Box είναι μια μέθοδος που χρησιμοποιείται για την εξέταση της λειτουργικότητας του λογισμικού χωρίς να γνωρίζει την εσωτερική δομή του κώδικα. Μπορεί να εφαρμοστεί σε όλα τα επίπεδα δοκιμών λογισμικού, αλλά χρησιμοποιείται κυρίως για τα υψηλότερα επίπεδα αποδοχής και σχετικά με το σύστημα. Πιο συγκεκριμένα, ένας επαγγελματίας που χρησιμοποιεί αυτήν τη μέθοδο για να ελέγξει τη λειτουργικότητα μιας εφαρμογής θα γνωρίζει μόνο την είσοδο και την αναμενόμενη έξοδο, αλλά όχι για το πρόγραμμα που βοηθά την εφαρμογή να φτάσει στην επιθυμητή έξοδο [34].

Σημαντικά πλεονεκτήματα του Black Box Testing:

- Αμερόληπτες δοκιμές επειδή ο σχεδιαστής και ο ελεγκτής λειτουργούν ανεξάρτητα.
- Ο ελεγκτής είναι απαλλαγμένος από κάθε πίεση γνώσης συγκεκριμένων γλωσσών προγραμματισμού για να ελέγξει την αξιοπιστία και τη λειτουργικότητα μιας εφαρμογής.

- Ο έλεγχος πραγματοποιείται από πλευράς του χρήστη και όχι από πλευράς σχεδιαστή/προγραμματιστή.

Μερικά από τα πλεονεκτήματα του Black Box Testing:

- Οι εκτιμήσεις μπορεί να είναι περιττές, εάν ήδη έχουν διαχειριστεί από τον σχεδιαστή/προγραμματιστή.
- Ο έλεγχος κάθε πιθανής ροής εισόδου δεν είναι δυνατός, επειδή είναι χρονοβόρος και αυτό θα αφήσει κομμάτια χωρίς να ελεγχθούν.
- Δεν μπορεί να χρησιμοποιηθεί για τη δοκιμή σύνθετων τμημάτων κώδικα.

Εικόνα 5.1 Γράφημα για μορφή του Black Box

5.1 Classifier

Ο στόχος του κεφαλαίου αυτού είναι να γίνει ανάλυση των κλειδιών από κάθε βιβλιοθήκη που έχουν δημιουργηθεί όπως αναφέρεται στο κεφάλαιο 3, για μελετήσουμε το ενδεχόμενο να μπορεί να αποκαλυφθεί η προέλευση ενός κλειδιού. Εάν κάποιος μπορεί να διαπιστώσει την προέλευση του κλειδιού δηλαδή να μπορεί να βρει από ποια βιβλιοθήκη κρυπτογράφησης προέρχεται, αυτό καθιστά τον αλγόριθμο ευάλωτο σε επιθέσεις καθώς επίσης οδηγεί στην έλλειψη ασφάλειας. Οι προαναφερθέντες κίνδυνοι μπορούν εύκολα να παρουσιαστούν αν από ένα κλειδί διαρρεύσουν πληροφορίες για τις επιλογές σχεδιασμού και υλοποίησης όπως τον πρωταρχικό αλγόριθμος παραγωγής του και από ποια βιβλιοθήκη προέρχεται.

Η μέθοδος ανάλυσης των κλειδιών που έγινε για σκοπούς της έρευνας αυτής είναι το Classification με την βοήθεια Μηχανικής Μάθησης. Η διαδικασία του Classification αλγορίθμου έχει ως εξής: Χρησιμοποιείται το σύνολο δεδομένων εκπαίδευσης για να

βρεθούν καλύτερες συνθήκες ορίου που θα μπορούσαν να χρησιμοποιηθούν για τον προσδιορισμό κάθε κατηγορίας στόχου. Μόλις καθοριστούν οι συνθήκες ορίου, η επόμενη εργασία είναι η πρόβλεψη της κατηγορίας στόχου.

5.1.1 Μηχανική Μάθηση

Η μηχανική εκμάθηση (ML) είναι η μελέτη αλγορίθμων υπολογιστών που βελτιώνεται αυτόματα μέσω της εμπειρίας [28]. Θεωρείται ως ένα υποσύνολο της τεχνητής νοημοσύνης. Οι αλγόριθμοι μηχανικής εκμάθησης δημιουργούν ένα μαθηματικό μοντέλο βασισμένο σε δείγματα δεδομένων, γνωστά ως "δεδομένα εκπαίδευσης", προκειμένου να λαμβάνουν προβλέψεις ή αποφάσεις χωρίς να έχουν προγραμματιστεί ρητά να το πράξουν.

5.1.2 Επιβλεπόμενη Μηχανική Μάθηση

Η πλειονότητα της πρακτικής μηχανικής μάθησης χρησιμοποιεί επιβλεπόμενη μάθηση. Έχοντας μεταβλητές εισόδου (x) και μεταβλητές εξόδου (Y), χρησιμοποιούνται από έναν αλγόριθμο Επιβλεπόμενης Μηχανικής Μάθησης για να μάθετε τη λειτουργία σύνδεσης από την είσοδο στην έξοδο Y = f (X). Ο στόχος είναι να προσεγγίσει τη λειτουργία σύνδεσης τόσο καλά ώστε όταν πάρει νέα δεδομένα εισόδου (x) να μπορεί να προβλέψει ή να κατηγοριοποιήσει τις μεταβλητές εξόδου (Y) για αυτά τα δεδομένα.

5.1.3 Εργαλείο Μηχανικής Μάθησης

Το εργαλείο που χρησιμοποιήθηκε για το κτίσιμο του Classifier είναι το Scikit-learn (sklearn) [29].
Το Scikit-learn είναι μια βιβλιοθήκη μηχανικής εκμάθησης για τη γλώσσα προγραμματισμού Python. Διαθέτει διάφορους αλγόριθμους classification, regression and clustering, συμπεριλαμβανομένων των vector machines, random forests, gradient boosting, k-means, και έχει σχεδιαστεί για να λειτουργεί με τις βιβλιοθήκες της Python, NumPy και SciPy.

Στην περίπτωση της έρευνας αυτής χρειάστηκαν μόνο οι αλγόριθμοι για classification.

5.1.4 Αλγόριθμοι Classification

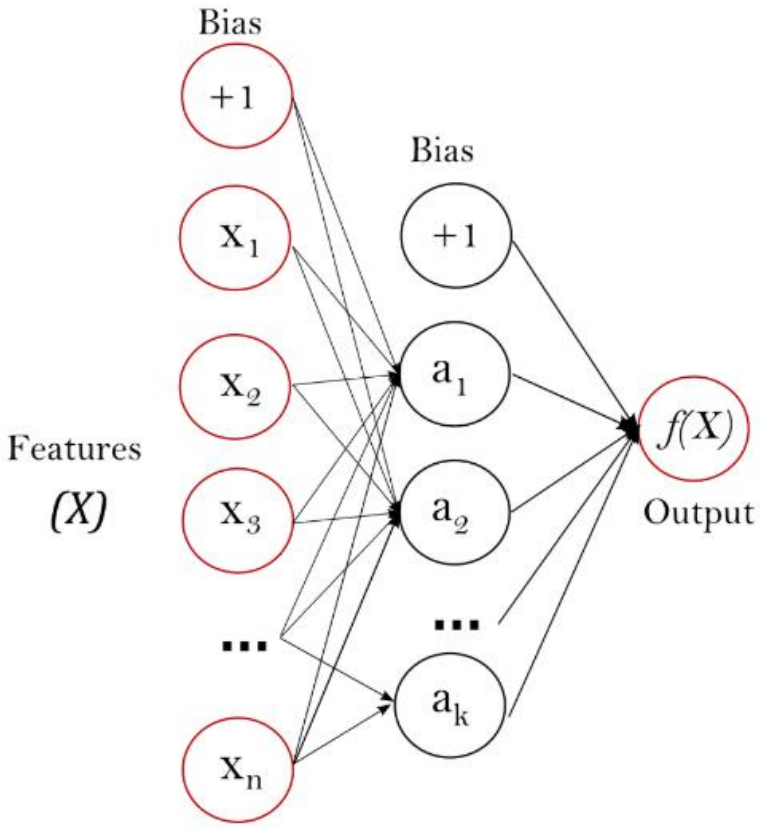
Υπάρχουν πάρα πολλοί αλγόριθμοι Classification σε αυτή την περίπτωση όμως οι αλγόριθμοι που χρησιμοποιήθηκαν είναι ο αλγόριθμος Decision Trees [30] Multi-layer Perceptron (MLP) classifier [31].

5.1.4.1 Decision Trees

Τα δέντρα απόφασης (DT) είναι μια μη παραμετρική επιβλεπόμενης μάθησης μέθοδος που χρησιμοποιείται για Classification και Regression. Ο στόχος είναι να δημιουργηθεί ένα μοντέλο που προβλέπει την τιμή μιας μεταβλητής στόχου μαθαίνοντας απλούς κανόνες απόφασης που συνάγονται από τις δυνατότητες δεδομένων.

5.1.4.2 Multi-layer Perceptron

Το Multi-layer Perceptron (MLP) είναι ένας αλγόριθμος επιβλεπόμενης μάθησης που μαθαίνει μια συνάρτηση $f(\cdot) : R^m \rightarrow R^o$ εκπαιδεύοντας σε ένα σύνολο δεδομένων, όπου m είναι ο αριθμός των διαστάσεων για την είσοδο και o είναι ο αριθμός των διαστάσεων για την έξοδο. Δίνεται ένα σύνολο χαρακτηριστικών $X = x_1, x_2, ..., x_m$ και ένας στόχος y, μπορεί να μάθει ένα μη γραμμικό εργαλείο προσέγγισης για Classification ή Regression. Είναι διαφορετικό από τη logistic regression, καθώς μεταξύ του επιπέδου εισόδου και εξόδου, μπορεί να υπάρχουν ένα ή περισσότερα μη γραμμικά layers, που ονομάζονται hidden layers. Το Εικόνα 5.2 δείχνει ένα hidden layer MLP με κλιμακωτή έξοδο.



Εικόνα 5.2 Hidden Layer MLP

5.1.5 Δημιουργία του Classifier

Για την δημιουργία του classifier, έχουν δημιουργηθεί δύο προγράμματα Python το classifierDT.py και το classifierMLP.py και βρίσκονται στο Παράρτημα Δ και Παράρτημα Ε αντίστοιχα. Με τους δυο αυτούς classifiers θέλουμε να δούμε κατά πόσο μπορεί ο αλγόριθμος παίρνοντας τα public και private keys να προβλέψει σε ποια από τις βιβλιοθήκες ανήκει.

Αρχικά τα δεδομένα πριν περάσουν στους δύο classifiers εξάγονται από την SQLite βάση δεδομένων σε μορφή csv αρχείου και προετοιμάζονται με την βοήθεια ενός προγράμματος γραμμένο σε java που βρίσκεται στο Παράρτημα Z και λέγεται DataPreprocessing.java. Το πρόγραμμα αυτό με απλά λόγια μετατρέπει τα δεδομένα σε μορφή την οποία θα μπορούν οι classifiers να αναγνωρίσουν. Πρώτο βήμα μετατρέπει τα κλειδιά από δεκαεξαδική μορφή σε δεκαδική μορφή και δεύτερο βήμα κωδικοποιεί τις βιβλιοθήκες, όπου 1 αντιστοιχεί στην OpenSSL, 2 στην NaCl και 3 στην BouncyCastle. Παράδειγμα του csv αρχείου μετά την προετοιμασία των δεδομένων φαίνεται στην Εικόνα 5.3.

	A	B	C
1	1.02E+76	3.09E+76	1
2	8.97E+76	5.44E+154	2
3	6.61E+76	9.53E+76	1
4	8.06E+76	5.98E+154	2
5	1.04E+77	6.47E+154	2
6	1.07E+77	6.54E+154	2
7	3.19E+76	1.03E+77	1

Εικόνα 5.3 Παράδειγμα από CSV αρχείο με κλειδία

Και τα δύο προγράμματα αρχικά διαβάζουν δεδομένα από CSV αρχείο της μορφής όπως φαίνεται στην Εικόνα 5.3. Στην πρώτη στήλη είναι τα public key, στην δεύτερη τα private keys και στην τρίτη στήλη η βιβλιοθήκη που χρησιμοποιήθηκε για την παραγωγή του κάθε ζεύγους κλειδιών σε κάθε γραμμή. Όπου 1 αντιστοιχεί στην OpenSSL, 2 στην NaCl και 3 στην BouncyCastle. Η πρώτες δύο στήλες είναι τα δεδομένα εισόδου και η τρίτη στήλη είναι τα δεδομένα εξόδου (expected output). Αφού διαβάσουν τα δεδομένα γίνεται η κανονικοποίηση τους με StandardScaler και το dataset περνά στον ανάλογο classifier.

Στην περίπτωση του classifierDT.py η μέθοδος που καλεί τον Decision Tree Classifier είναι η DecisionTreeClassifier(max_leaf_nodes=3, criterion='entropy', random_state=0), όπου παίρνει ως παραμέτρους:

- max_leaf_nodes που αναπτύσσει ένα δέντρο με max_leaf_nodes με best-first λογική για μείωση της ακαθαρσίας των δεδομένων
- criterion='entropy'που είναι η συνάρτηση μέτρησης της ποιότητας ενός διαχωρισμού. Το κριτήριο είναι η εντροπία.
- random_state=0 που ελέγχει την τυχαιότητα του estimator.

Στην περίπτωση του classifierMLP.py η μέθοδος που καλεί τον Decision Tree Classifier είναι η MLPClassifier(solver='sgd', alpha=1e-5, hidden_layer_sizes=(3, 3), random_state=1, max_iter=3000), όπου παίρνει ως παραμέτρους:

- solver='sgd' που είναι ο επιλυτής για βελτιστοποίηση βαρών. Το 'sgd' αναφέρεται στη στοχαστική κατάβαση κλίσης.
- alpha=1e-5 που είναι ο όρος regularization
- hidden_layer_sizes=(3, 3) Οι αριθμοί των νευρώνων στα κρυφά επίπεδα
- random_state=1 που ελέγχει την τυχαιότητα του estimator

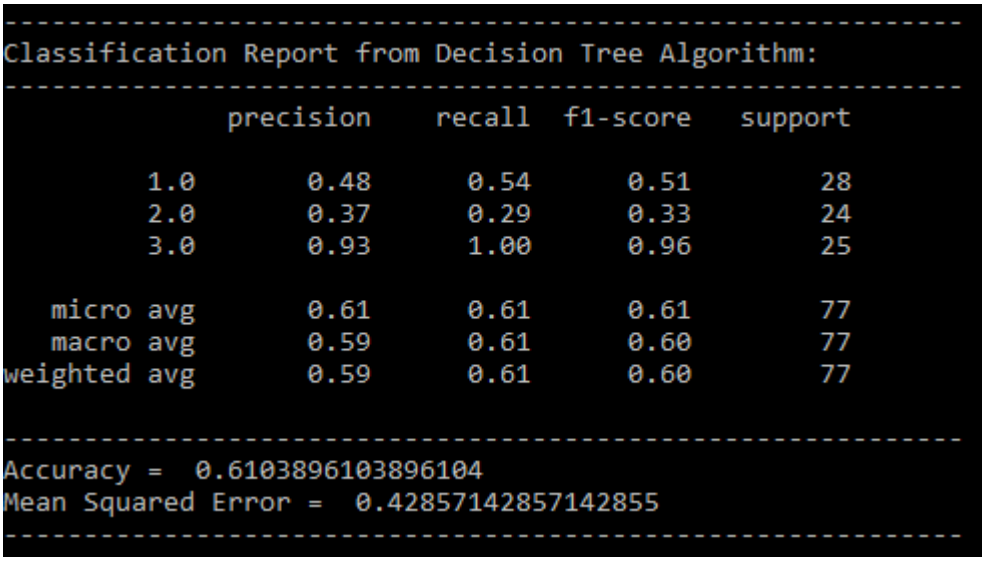
- max_iter=3000 που είναι ο αριθμός των εποχών (πόσες φορές θα χρησιμοποιηθεί κάθε δεδομένο)

Στην συνέχεια και στα δυο προγράμματα γίνεται η προσαρμογή των training data με τη βοήθεια της μεθόδου fit() ούτως ώστε να μάθει το μοντέλο. Δηλαδή να προσαρμοστούν για να γίνει η πρόβλεψη στα testing data. Μετά το fit, γίνεται η πρόβλεψη με την βοήθεια της συνάρτησης predict() που παίρνει ως παράμετρο τα test data και προβλέπει τα πιθανά αποτελέσματα.

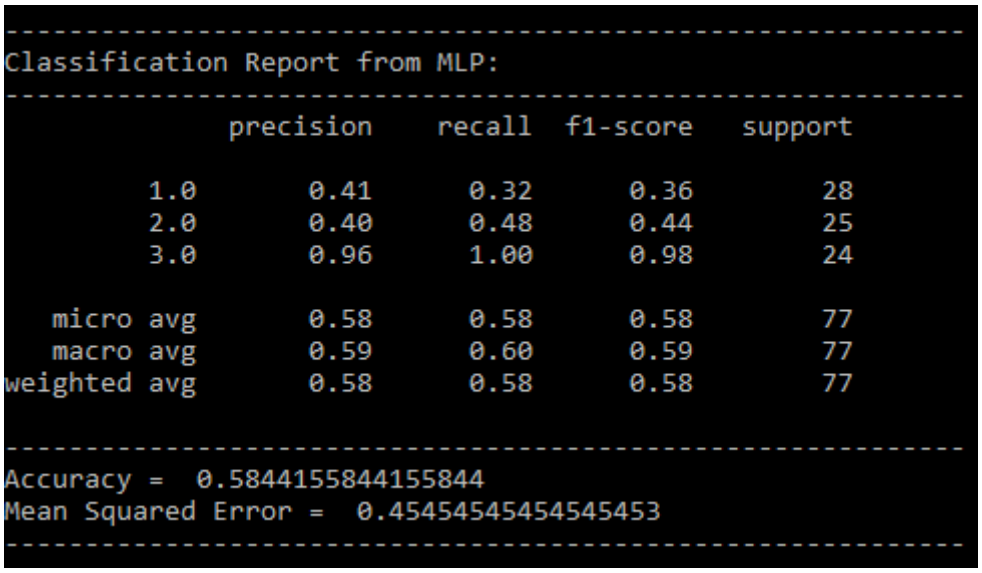
Τέλος, τα δύο προγράμματα υπολογίζουν το score της κάθε πρόβλεψης με την βοήθεια της μεθόδου classification_report(). Η μέθοδος classification_report() εμφανίζει το precision, το recall και το f1-score. Η ακρίβεια δηλαδή το precision είναι η ικανότητα του Classifier να μην επισημαίνει ως θετικό ένα δείγμα που είναι αρνητικό και η καλύτερη τιμή του είναι 1 και η χειρότερη τιμή του είναι 0. Το recall είναι η ικανότητα του Classifier να βρίσκει όλα τα θετικά δείγματα και η καλύτερη τιμή του είναι 1 και η χειρότερη τιμή του είναι 0. Και το f1-score είναι ο σταθμισμένος μέσος όρος του precision και του recall και η καλύτερη τιμή του είναι 1 και η χειρότερη τιμή του είναι 0.

5.2 Αποτελέσματα

Τρέχοντας τα δύο Python προγράμματα που περιεγράφηκαν στο κεφάλαιο 5.1.5, τα αποτελέσματα δεν ήταν αρκετά ικανοποιητικά παρόλο που οι Classifiers είχαν εκπαιδευτεί σωστά. Όπως φαίνεται παρακάτω στην Εικόνα 5.4 και στην Εικόνα 5.5 είναι παραδείγματα εκτέλεσης των προγραμμάτων classifierDT.py και classifierMLP.py αντίστοιχα. Είναι φανερό από τις δύο εικόνες ότι τα αποτελέσματα και των δύο προγραμμάτων έχουν πολύ κοντινά αποτελέσματα.



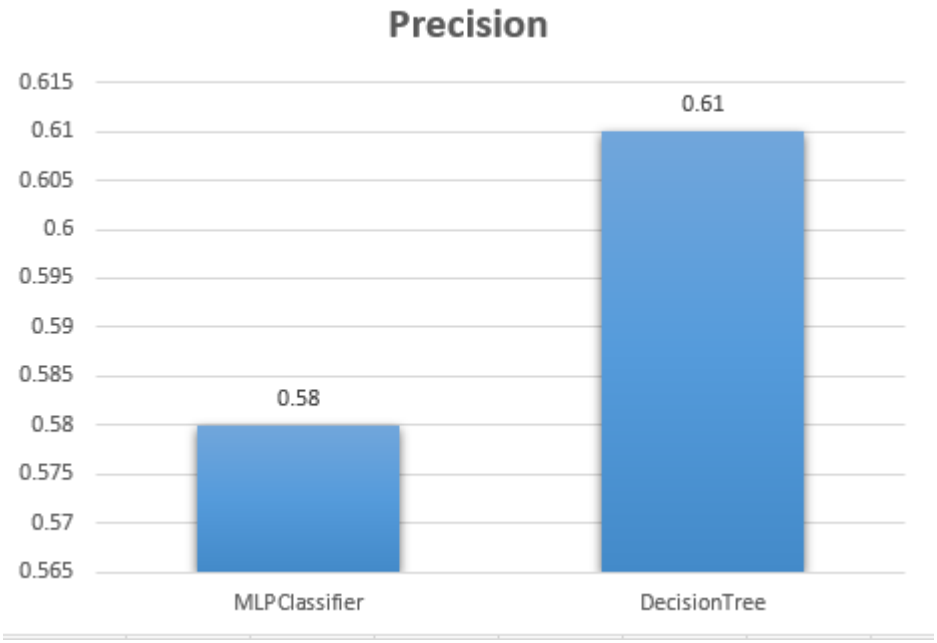
Εικόνα 5.4 Παράδειγμα εκτέλεσης του classifierDT.py



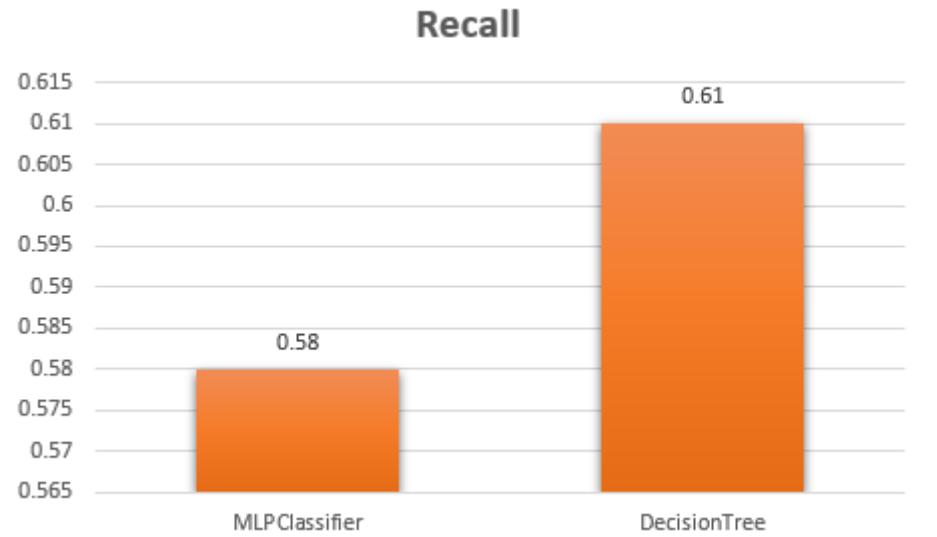
Εικόνα 5.5 Παράδειγμα εκτέλεσης του classifierMLP.py

Με βάση τα αποτελέσματα τις εικόνας παρόλο που το γενικό accuracy είναι λίγο πιο πάνω από το 50%, μόνο το precision του 3 είναι ψηλό. Όπου 1 αντιστοιχεί στην OpenSSL, 2 στην NaCl και 3 στην BouncyCastle. Που σημαίνει ότι κατάφεραν και οι δύο classifiers να προβλέψουν σχετικά καλά από που προήλθαν τα κλειδιά της βιβλιοθήκης BouncyCastle. Το ίδιο ισχύει και για τις τιμές του recall και του f1-score για το BouncyCastle. Οι υπόλοιπες μετρικές για το OpenSSL και το NaCl είναι πολύ χαμηλές που δείχνουν ότι εκεί οι classifiers δεν κατάφεραν να φτάσουν πολύ κοντά στο πραγματικό αποτέλεσμα.

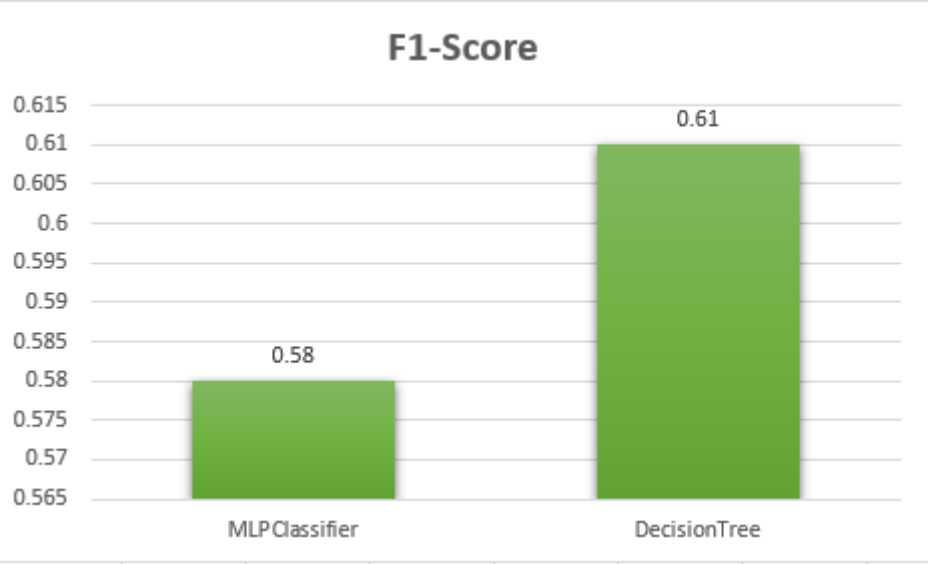
Τρέχοντας για αρκετές φορές τους classifiers παράχθηκαν οι γραφικές παραστάσεις Γράφημα 5.1, Γράφημα 5.2, και Γράφημα 5.3, παίρνοντας τον μέσο όρο των τιμών για precision, recall και f1-score.



Γράφημα 5.1 Γραφική παράστας μέσου όρου του precision για MLP και DecisionTree



Γράφημα 5.2 Γραφική παράστας μέσου όρου του recall για MLP και DecisionTree



Γράφημα 5.3 Γραφική παράστας μέσου όρου του F1-Score για MLP και DecisionTree

Παρατηρώντας τις πιο πάνω γραφικές παραστάσεις φαίνεται ότι ο Decision Tree Classifier έχει λίγο καλύτερα αποτελέσματα από τον MLP. Σε γενικές γραμμές όμως κανένας από τους δύο αλγορίθμους μπόρεσε να δώσει καλά αποτελέσματα τα οποία να αποδεικνύουν ότι από τα κλειδιά μπορούμε να ανακαλύψουμε την προέλευση τους. Συμπεραίνοντας, ίσως τα κλειδιά να μην αποκαλύπτουν κάποια σχέση την οποία να μπορεί να εντοπίσει κάποιος εκπαιδευμένος Classifier και αυτό από την μία πλευρά καθιστά τις βιβλιοθήκες OpenSSL, NaCl και BouncyCastle ασφαλείς ως προς την τυχαιότητα των κλειδιών που παράγουν. Από την άλλη πλευρά όμως, αφού τα αποτελέσματα από αυτούς τους classifiers σε κάποιες περιπτώσεις ήταν καλά, σημαίνει ότι θα μπορούσε ένα καλύτερα κτισμένο νευρωνικό δίκτυο να εντοπίσει πιθανές σχέσεις μεταξύ των κλειδιών. Αυτό θα αποτελούσε μεγάλο πλεονέκτημα για επιτήδειους εισβολείς κάποιου κρυπτογραφικού συστήματος που θα μπορούσε να αποκαλύψει την προέλευση και την λειτουργία του μόνο από τα κλειδιά που παράγει.

Κεφάλαιο 6

Συμπεράσματα

6.1 Πλεονεκτήματα Elliptic Curve Κρυπτογραφίας

Το κύριο πλεονέκτημα της χρήσης κρυπτογραφίας Elliptic Curve είναι το μειωμένο μέγεθος κλειδιού και συνεπώς η ταχύτητα. Οι αλγόριθμοι που βασίζονται σε ελλειπτική καμπύλη χρησιμοποιούν σημαντικά μικρότερα μεγέθη κλειδιών από τα αντίστοιχα μη ελλειπτικών καμπυλών. Η διαφορά στα ισοδύναμα μεγέθη κλειδιών αυξάνεται δραματικά καθώς αυξάνονται τα μεγέθη κλειδιών. Η κατά προσέγγιση αντιστοιχία της ασφαλείας για συμμετρικούς αλγόριθμους σε σύγκριση με τους τυπικούς ασύμμετρους αλγόριθμους και τους Elliptic Curve αλγορίθμους φαίνεται στον πίνακα της Εικόνας 6.1.

Symmetric Key Length	Standard asymmetric Key Length	Elliptic Curve Key Length
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Εικόνα 6.1 Σύγκριση Συμμετρικών αλγόριθμων, Ασύμμετρων αλγορίθμων και Elliptic Curve αλγορίθμων

Όπως φαίνεται, ένας τυπικός ασύμμετρος αλγόριθμος για να έχει ισοδύναμη ισχύ με ένα συμμετρικό κλειδί 256 bit θα πρέπει να χρησιμοποιήσει ένα τεράστιο κλειδί 15360 bit. Τα

κλειδιά αυτού του μεγέθους συνήθως δεν είναι πρακτικά λόγω της επεξεργαστικής δύναμης που απαιτείται και συνεπώς, της ταχύτητας των εργασιών.

6.2 Επίλογος

Παρά τη αξιοσημείωτη συζήτηση σχετικά με το εάν υπάρχει backdoors προς την γεννήτρια τυχαίων αριθμών ελλειπτικής καμπύλης, ο αλγόριθμος, στο σύνολό του, παραμένει αρκετά ασφαλής. Παρόλο που υπάρχουν πολλές δημοφιλείς ευπάθειες στις επιθέσεις side-channel [35], μετριάζονται εύκολα μέσω πολλών τεχνικών. Επιπλέον, παρόλο που τα μεγαλύτερα κλειδιά ECC σπάνε δημόσια κάθε τόσο, το ίδιο ισχύει και για όλους τους άλλους δημοφιλείς τύπους αλγορίθμων. Όμως, ανεξάρτητα από το πόσο ασφαλές είναι το ECC, θεωρητικά, η εφαρμογή του πρέπει να γίνεται σωστά ώστε να μειώνεται η πιθανότητα επιτυχών επιθέσεων. Η ιστορία έχει δείξει ότι κάτι τέτοιο είναι κρίσιμο, καθώς μεγάλες ομάδες και εταιρείες απέτυχαν για τον λόγο αυτό. Σύμφωνα και με τους προαναφερθείς ελέγχους που έγιναν για σκοπούς αυτής της έρευνας, άλλα και για άλλους μελλοντικούς, υπογραμμίζεται η αναγκαιότητα του σωστού ελέγχου τόσο της ασφάλειας όσο και της σωστής εφαρμογής του αλγορίθμου Elliptic Curves.

Τέλος, παρόλο που το τέλειο, ευρέως εφαρμόσιμο και άθραυστο κρυπτοσύστημα δεν υπάρχει, υπάρχουν πολλοί τρόποι για να διατηρηθούν τα δεδομένα ασφαλή. Υπάρχει μια ποικιλία κατηγοριών κρυπτοαλγορίθμων, συμπεριλαμβανομένων αλγορίθμων κατακερματισμού, συμμετρικών κρυπτοαλγορίθμων και ασύμμετρων κρυπτοαλγορίθμων. Το ECC, όπως και το RSA, εμπίπτει στην ταξινόμηση ασύμμετρου αλγορίθμου. Αυτός ο τύπος κρυπτογράφου ECC επιλύει μια ποικιλία προβλημάτων, ένα από τα οποία επιτρέπει σε δύο κόμβους ή άτομα που δεν έχουν επικοινωνήσει ποτέ μεταξύ τους πριν να μεταφέρουν πληροφορίες ο ένας στον άλλο με ασφαλή τρόπο. Αυτοί οι αλγόριθμοι είναι επίσης κρίσιμοι στον μηχανισμό πολλών πρωτοκόλλων, προτύπων, υπηρεσιών και υποδομών. Η σημαντικότητα τους άλλωστε είναι ολοφάνερη αφού οι πλείστες εφαρμογές τόσο του σήμερα όσο και του αύριο, χρησιμοποιούν τις τεχνολογίες αυτές για να διασφαλίσουν την προστασία κάθε είδους δεδομένων και πληροφοριών. Λόγω της ραγδαίας εξέλιξης νέων εφαρμογών και μέσων, η αναγκαιότητα για την βέλτιστη λειτουργία των τεχνολογιών κρυπτογράφησης είναι άκρως σημαντική.

Βιβλιογραφία

[1] Stallings, William. (2003). Cryptography and Network Security: Principles And Practices

[2] Darrel Hankerson and Alfred Menezes. Elliptic curve cryptography. Springer, 2011

[3] Cathalo, Julien & Petit, Christophe. (2010). One-Time Trapdoor One-Way Functions. 6531. 283-298. 10.1007/978-3-642-18178-8_25

[4] Washington, L.C.. (2008). Elliptic curves: Number theory and cryptography, second edition.

[5] OpenSSL Foundation, I., n.d. /Index.Html. [online] Openssl.org. Available at: <https://www.openssl.org/>

[6] Weisstein, Eric W. "Irreducible Polynomial." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/IrreduciblePolynomial.html

[7] Certicom Research, http://www.secg.org/sec2-v2.pdf

[8] https://www.sqlite.org/about.html. About SQLite. (n.d.). SQLite Home Page. https://www.sqlite.org/about.html

[9] NaCl Introduction. https://nacl.cr.yp.to/

[10] Internals. Introduction. https://nacl.cr.yp.to/internals.html

[11] PyNaCl. PyPI. https://pypi.org/project/PyNaCl/

[12] Adam Langley, Mike Hamburg, and Sean Turner. Elliptic curves for security. RFC7748, January 2016

[13] Bernstein. "Irrelevant patents on elliptic-curve cryptography". cr.yp.to. Retrieved 2016-02-08

[14] The bouncy castle crypto package is a java implementation of cryptographic algorithms. http://www.bouncycastle.org/

[15] Org.bouncycastle.math.ec (2014, July 28). Javadoc made better by java experts colaboration. https://javadox.com/org.bouncycastle/bcprov-jdk15on/1.51/org/bouncycastle/math/ec/package-summary.html

[16] Elliptic curve key pair generation and key factories - Java APIs 1.X - The legion of the bouncy castle. (n.d.). bouncycastle.org. https://www.bouncycastle.org/wiki/display/JA1/Elliptic+Curve+Key+Pair+Generation+and+Key+Factories

[17] Williams, Laurie. "White-Box Testing" (PDF): 60–61, 69. Retrieved 13 February 2013

[18] Ammann, Paul; Offutt, Jeff (2008). Introduction to Software Testing. Cambridge University Press. ISBN 9780521880381

[19] OpenSSL: Manual page of RAND https://www.openssl.org/docs/crypto/RAND. html

[20] P. Lacharme, A. R¨ock, V. Strubel, M. Videau: The Linux Pseudorandom Number Generator Revisited (2012) http://eprint.iacr.org/2012/251.pdf.

[21] Strenzke, F. (2016). An analysis of OpenSSL’s random number generator. Advances in Cryptology – EUROCRYPT 2016, 644-669. https://doi.org/10.1007/978-3-662-49890-3_25

[22] Strenzke, F. (2016). An analysis of OpenSSL’s random number generator. Advances in Cryptology – EUROCRYPT 2016, 644-669. https://doi.org/10.1007/978-3-662-49890-3_25

[23] Daniel J Bernstein. Extending the Salsa20 nonce. In Workshop record of Symmetric Key Encryption Workshop, volume 2011, 2011 [Adam Langley and Yoav Nir. ChaCha20 and Poly1305 for IETF protocols. 2015.

[24] Libsodium v1.0.12 and v1.0.13 security assessment. (2017, December 15). Private Internet Access Blog. https://www.privateinternetaccess.com/blog/libsodium-v1-0-12-and-v1-0-13-security-assessment/

[25] The bouncy castle crypto package is a java implementation of cryptographic algorithms. http://www.bouncycastle.org/

[26] Org.bouncycastle.crypto.prng (Bouncy castle library 1.37 API specification). (2007, June 14). People @ EECS at UC Berkeley. https://people.eecs.berkeley.edu/~jonah/bc/org/bouncycastle/crypto/prng/package-summary.html

[27] Sylvain Ruhault. Security analysis for pseudo-random number generators. Cryptography and Security [cs.CR]. Ecole normale supérieure - ENS PARIS, 2015. English. ffNNT : 2015ENSU0014ff. fftel-01236602v2

[28] Mitchell, T. M. (1997). Machine learning.

[29] Scikit-learn. (n.d.). scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation, from https://scikit-learn.org/stable/

[30] 1.10. Decision trees — scikit-learn 0.23.1 documentation. (n.d.). scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. from https://scikit-learn.org/stable/modules/tree.html

[31] Sklearn.neural_network.MLPClassifier — scikit-learn 0.23.1 documentation. (n.d.). scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. from https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[32] Digital signature standard (dss), fips pub 186-2 with change notice. National Institute of Standards and Technology (NIST), FIPS PUB 186-2, U.S. Department of Commerce, January 2000

[33] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators

[34] Jerry Gao; H.-S. J. Tsao; Ye Wu (2003). Testing and Quality Assurance for Component-based Software. Artech House. pp. 170–. ISBN 978-1-58053-735-3

[35] Bar-El, H.; "Introduction to Side Channel Attacks," Discretix Technologies Ltd., 2003

ΠΑΡΑΡΤΗΜΑ Α

```
import os
import pem
import subprocess
import sqlite3
#
================================================================================
==============  #
#       This is a script that generates 1 million EC key pairs with
OpenSSL library           #
#
================================================================================
==============  #
#Create connection with the SQLite DB
conn = sqlite3.connect('dbEC.sqlite')
cur = conn.cursor()
numOfKeyPairs = 1000000

#Generate 1 million openSSL EC key pairs
for x in range(numOfKeyPairs):
      print("Key Pair number: ", x);
      genPrivate256 = 'openssl ecparam -name secp256k1 -genkey -noout -out
private.pem'
      genPublic256 = 'openssl ec -in private.pem -pubout -out public.pem'
      os.system(genPrivate256)
      os.system(genPublic256)

      genHex = 'openssl ec -in private.pem -text -noout > hex.txt'
      os.system(genHex)

      #The following code reads the pem files as txt files and takes only
the hex key as a string
      f = open("hex.txt", "r")
      f.readline()
      f.readline()
      privateHex = f.readline() + f.readline() + f.readline()
      privateHex = privateHex.replace(" ", "")
```

```
      privateHex = privateHex.replace("\n", "")
      privateHex = privateHex.replace(":", "")

      f.readline()
      publicHex = f.readline() + f.readline() + f.readline() +
f.readline() + f.readline()
      publicHex = publicHex.replace(" ", "")
      publicHex = publicHex.replace("\n", "")
      publicHex = publicHex.replace(":", "")

      f.close()

      # Here is the prepared query statement that inserts the hex keys in
the SQLite DB
      cur.execute('INSERT INTO KEYS (Framework, PrivateKey, PublicKey,
Metadata) values (?, ?, ?, ?)',
            ('OpenSSL', privateHex, publicHex, 'secp256k1'))
      conn.commit()

      #Remove all created files that are unnecessary
      rmPBpem = 'rm public.pem'
      os.system(rmPBpem)
      rmPRpem = 'rm private.pem'
      os.system(rmPRpem)

      rmHexTxt = 'rm hex.txt'
      os.system(rmHexTxt)
conn.close()
```

```
from nacl.public import PrivateKey
import binascii
import os
import pem
import subprocess
import sqlite3
#
================================================================================
==============  #
#     This is a script that generates 1 million EC key pairs with
Nacl/libSodium library      #
#                                       Curve25519
#
#
================================================================================
==============  #
#Create connection with the SQLite DB
conn = sqlite3.connect('dbEC.sqlite')
cur = conn.cursor()

numOfKeyPairs = 1000000
#Generate 1 million openSSL EC key pairs
for x in range(numOfKeyPairs):
      print("Key Pair number: ", x);
      privKey = PrivateKey.generate()
      pubKey = privKey.public_key

      private = binascii.hexlify(bytes(privKey))
      public = binascii.hexlify(bytes(pubKey))

      # Here is the prepared query statement that inserts the hex keys in
the SQLite DB
      cur.execute('INSERT INTO KEYS (Framework, PrivateKey, PublicKey,
Metadata) values (?, ?, ?, ?)',
            ('Nacl', private, public, 'Curve25519'))
      conn.commit()

conn.close()
```

```
import java.io.FileWriter;
import java.math.BigInteger;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Security;
import java.util.ArrayList;
import java.util.List;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import java.security.spec.ECGenParameterSpec;
/**
 * The following program generates an EC key pair from the curve secp256k1
using
 * the Bouncy Castle library. The key pair is inserted in a csv file for
further
 * use.
 *
 * The extraction of the actual hex value of the private key is not
possible at
 * the moment because java does not allow showing any sensitive content
such as
 * the private key. As a result, only the public keys are saved in the csv
file.
 *
 * @author Christina Procopiou
 *
 */
public class GenEC_BouncyCastle {

      public static void main(String[] args) {

            Security.addProvider(new BouncyCastleProvider());
            System.out.println("Starting...");
            String name = "secp256k1"; // secp256k1 curve
            try {
                  //Generate 1 million BouncyCastle EC key pairs
                  int numOfKeyPairs = 1000000;
                  FileWriter csvWriter = new FileWriter("bcPublic.csv");
                  for (int i = 0; i < numOfKeyPairs; i++) {
                        System.out.println("Num of key:" + i);
                        KeyPairGenerator kpg =
KeyPairGenerator.getInstance("ECDSA", BouncyCastleProvider.PROVIDER_NAME);
                        kpg.initialize(new ECGenParameterSpec(name));
                        KeyPair keyPair = kpg.generateKeyPair();

                        PublicKey pub = keyPair.getPublic();
                        PrivateKey prv = keyPair.getPrivate(); // not used

                        String publicString = pub.toString();

                        System.out.println(publicString);
                        String publicKeyStr = (String)
publicString.subSequence(14, 74);
```

```
                        String hex = publicKeyStr.replace(":",
"").replace("[", "").replace("]", "");
                        System.out.println(hex);
                        BigInteger pbk = new BigInteger(hex, 16);
                        System.out.println(pbk);

                        List<String> rowData = new ArrayList<String>();

                        rowData.add(pbk.toString());

                        rowData.add("3");

                        csvWriter.append(String.join(",", rowData));
                        csvWriter.append("\n");
                  }
                  csvWriter.flush();
                  csvWriter.close();

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

```
#
=========================================================================
======================== #
#                IMPORT OF THE NECESSARY LIBRARIES REQUIRED FOR THIS
RESEARCH              #
#
=========================================================================
======================== #
#       System's required libraries (os, sys,csv)
import os
import sys
import csv
#
=========================================================================
======================== #
#                  pandas: is a software library for data manipulation and analysis#
#
=========================================================================
======================== #
#                  Scikit-learn is a free machine learning library for Python.                  #
#
=========================================================================
======================== #
import pandas as pd
import numpy as np
import sklearn as sk
#
=========================================================================
======================== #
#                  Classifier import, splitting and clasSifying report matrix                  #
#
=========================================================================
======================== #
from sklearn import tree
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error

#
================================================================
====================== #
#  Generation of the appropriate dataframe (in pandas) for a given dataset (csv)          #
#
================================================================
====================== #
def read_Dataset(filename):
    dt = pd.read_csv(str(filename), sep=',').values
    dt = pd.DataFrame(dt)
    dt = dt.values
    dt = np.nan_to_num(dt)
    return dt

#   Call function that reads the appropriate csv (for dataset) into a panda dataframe dataset
variable
dataset = read_Dataset(sys.argv[1])

#   Split Dataset into attribute vectors (X) and class attribute (y)
X = dataset[:,:-1]    #  X Features are stored in the first elements of the dataset
Y = dataset[:,len(dataset[0])-1]          #  Y Label instance is the last value of the dataset
#  Splitting dataset into training and testing sets by using a percentage of sys.argv[2]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = float(sys.argv[2]))

estimatorDT = Pipeline([("scale", StandardScaler()),
    ("estimatorDT", DecisionTreeClassifier(max_leaf_nodes=3, criterion='entropy',
random_state=0))])
```

```
estimatorDT.fit(X_train, y_train)

#Predict test data
y_pred_DT = estimatorDT.predict(X_test)

#Presentation of results
print('\n-------------------------------------------------------------');
print('Classification Report from Decision Tree Algorithm:')
print('-------------------------------------------------------------');
print(classification_report(y_test, y_pred_DT))
print('-------------------------------------------------------------');
print("Accuracy = ", accuracy_score(y_test, y_pred_DT))
print("Mean Squared Error = ", mean_squared_error(y_test, y_pred_DT))
print('-------------------------------------------------------------');
```

```
#
=============================================================================
======================== #
#                IMPORT OF THE NECESSARY LIBRARIES REQUIRED FOR THIS
RESEARCH              #
#
=============================================================================
======================== #
#       System's required libraries (os, sys,csv)
import os
import sys
import csv
#
=============================================================================
======================== #
#                  pandas: is a software library for data manipulation and analysis#
#
=============================================================================
======================== #
#                  Scikit-learn is a free machine learning library for Python.                  #
#
=============================================================================
======================== #
import pandas as pd
import numpy as np
import sklearn as sk
#
=============================================================================
======================== #
#                  Classifier import, splitting and clasSifying report matrix                  #
#
=============================================================================
======================== #
from sklearn.neural_network import MLPClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neural_network import MLPClassifier

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
#
================================================================
====================== #
#  Generation of the appropriate dataframe (in pandas) for a given dataset (csv)          #
#
================================================================
====================== #
def read_Dataset(filename):
    dt = pd.read_csv(str(filename), sep=',').values
    dt = pd.DataFrame(dt)
    dt = dt.values
    dt = np.nan_to_num(dt)
    return dt

#   Call function that reads the appropriate csv (for dataset) into a panda dataframe dataset
variable
dataset = read_Dataset(sys.argv[1])

#   Split Dataset into attribute vectors (X) and class attribute (y)
X = dataset[:,:-1]    #  X Features are stored in the first elements of the dataset
Y = dataset[:,len(dataset[0])-1]          #  Y Label instance is the last value of the dataset
#  Splitting dataset into training and testing sets by using a percentage of sys.argv[2]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = float(sys.argv[2]))

estimatorMLP = Pipeline([("scale", StandardScaler()),
    ("estimatorMLP", MLPClassifier(solver='sgd', alpha=1e-5, hidden_layer_sizes=(3, 3),
random_state=1, max_iter=3000))])
estimatorMLP.fit(X_train, y_train)
```

```
#Predict test data
y_pred_MLP = estimatorMLP.predict(X_test)

#Presentation of results
print('\n-------------------------------------------------------------');
print('Classification Report from MLP: ')
print('-------------------------------------------------------------');
print(classification_report(y_test, y_pred_MLP))
print('-------------------------------------------------------------');
print("Accuracy = ", accuracy_score(y_test, y_pred_MLP))
print("Mean Squared Error = ", mean_squared_error(y_test, y_pred_MLP))
print('-------------------------------------------------------------');
```

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.math.BigInteger;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;

/**
 * The following program takes the contents of a csv file and
prepares the data to fit
 * them later in a Neural Network classifier.
 * The data extracted from the csv file are prepared and are
converted from hexadecimals
 * to decimals because the Neural Network classifiers do not accept
hexadecimals
 *
 * Moreover, for each crypto library name there is a code:
 * OpenSSL is 1
 * Nacl is 2
 * BouncyCastle is 3
 *
 * @author Christina Procopiou
 *
 */
public class DataPreprocessing {


      public static void main(String[] args) {

            Path currentRelativePath = Paths.get("");
            String path =
currentRelativePath.toAbsolutePath().normalize().toString();
            path = path.replaceAll("\\\\", "//");
            path += "//";
            String csvFile = path + "KEYS.csv";
            BufferedReader br = null;
            String line = "";
            String cvsSplitBy = ",";
            int c = 0;
            try {
                  FileWriter csvWriter = new FileWriter("new.csv");

                  br = new BufferedReader(new FileReader(csvFile));
                  while ((line = br.readLine()) != null) {
                        System.out.println("Key = " + c);
                        // use comma as separator
                        String[] keys = line.split(cvsSplitBy);
                        List<String> rowData = new
ArrayList<String>();
                              if (keys[0].contains("Nacl")) {
                                    rowData.add("1");
                              } else if (keys[0].contains("OpenSSL")) {
```

```
                                    rowData.add("2");
                               } else if
(keys[0].contains("BouncyCastle")) {
                                    rowData.add("3");
                               }

                               BigInteger pbk = new BigInteger(keys[1],
16);
                               rowData.add(pbk.toString());
                               BigInteger prk = new BigInteger(keys[2],
16);
                               rowData.add(prk.toString());

                          csvWriter.append(String.join(",", rowData));
                          csvWriter.append("\n");
                          c++;
                     }
                     csvWriter.flush();
                     csvWriter.close();

               } catch (FileNotFoundException e) {
                     e.printStackTrace();
               } catch (IOException e) {
                     e.printStackTrace();
               } finally {
                     if (br != null) {
                          try {
                               br.close();
                          } catch (IOException e) {
                               e.printStackTrace();
                          }
                     }
               }
          }
}
```