ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΔΙΙΔΡΥΜΑΤΙΚΟ ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΦΑΡΜΟΣΜΕΝΗ ΕΠΙΧΕΙΡΗΣΙΑΚΗ ΕΡΕΥΝΑ ΚΑΙ ΑΝΑΛΥΣΗ

ΣΤΡΑΤΙΩΤΙΚΗ ΣΧΟΛΗ ΕΥΕΛΠΙΔΩΝ
Τμήμα Στρατιωτικών Επιστημών

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
Σχολη Μηχανικών Παραγωγής και Διοίκησης

(ΠΔ 97 /2015/ΦΕΚ 163Α/20.08.2014)

ECDSA
Η Ψηφιακή Υπογραφή της NSA

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

του

ΑΝΑΣΤΑΣΙΟΥ Χ. ΣΕΡΕΤΙΔΗ
ΑΜ: 2014018047

Επιβλέπων: Δημήτριος Πουλάκης
Καθηγητής Α.Π.Θ

Αθήνα, Απρίλιος 2017

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΔΙΙΔΡΥΜΑΤΙΚΟ ΔΙΑΤΜΗΜΑΤΙΚΟ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΦΑΡΜΟΣΜΕΝΗ ΕΠΙΧΕΙΡΗΣΙΑΚΗ ΕΡΕΥΝΑ ΚΑΙ ΑΝΑΛΥΣΗ
(ΠΔ 97 /2015/ΦΕΚ 163Α΄/20.08.2014)

ECDSA
Η Ψηφιακή Υπογραφή της NSA

Μεταπτυχιακη Διατριβη

του

ΑΝΑΣΤΑΣΙΟΥ Χ. ΣΕΡΕΤΙΔΗ
ΑΜ: 2014018017

Επιβλέπων: Δημήτριος Πουλάκης
Καθηγητής Α.Π.Θ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .....

.....	.....	.....
Δημήτριος Πουλάκης	Νικόλαος Ματσατσίνης	Νικόλαος Μπάρδης
Καθηγητής Α.Π.Θ	Καθηγητής Π.Κ	Αν.Καθηγητής Σ.Σ.Ε

Αθήνα, Απρίλιος 2017

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΔΙΙΔΡΥΜΑΤΙΚΟ ΔΙΑΤΜΗΜΑΤΙΚΟ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΦΑΡΜΟΣΜΕΝΗ ΕΠΙΧΕΙΡΗΣΙΑΚΗ ΕΡΕΥΝΑ ΚΑΙ ΑΝΑΛΥΣΗ
(ΠΔ 97 /2015/ΦΕΚ 163Α΄/20.08.2014)

Υπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της μεταπτυχιακής διατριβής, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην μεταπτυχιακή διατριβή. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων. Επίσης, βεβαιώνω ότι αυτή η μεταπτυχιακή διατριβή στην Εφαρμοσμένη Επιχειρησιακή Έρευνα και Ανάλυση, προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του Διιδρυματικού Διατμηματικού Προγράμματος Μεταπτυχιακών Σπουδών του Τμήματος Στρατιωτικών Επιστημών της Στρατιωτικής Σχολής Ευελπίδων και της Σχολής Μηχανικών Παραγωγής και Διοίκησης του Πολυτεχνείου Κρήτης.

.....
Αναστάσιος Χ.Σερετίδης
Πτυχιούχος Μαθηματικός Πανεπιστημίου Ιωαννίνων

στους γονείς μου

Περιεχόμενα

Περίληψη

Ο Αλγόριθμος Ψηφιακής Υπογραφής Ελλειπτικής Καμπύλης (ECDSA) είναι ένα ευρέως γνωστό σχήμα ψηφιακής υπογραφής, το οποίο χρησιμοποιείται από μεγάλους οργανισμούς, συμπεριλαμβανομένης της Υπηρεσίας Εθνικής Ασφάλειας (NSA) των ΗΠΑ, για την πιστοποίηση της προέλευσης των διακινούμενων δεδομένων. Ο μηχανισμός λειτουργίας της αυθεντικότητας των μηνυμάτων που υπογράφονται με το σχήμα αυτό, συνδυάζει ιστορικά δυσεπίλυτα μαθηματικά προβλήματα, που πέρα από την αισθητική τους αξία, καθιστούν τον όρο «ψηφιακή υπογραφή» ένα από τα σημαντικότερα επιτεύγματα της επιστήμης υπολογιστών.

Ο ευαίσθητος τομέας της ασφάλειας της πληροφορίας τόσο στην διεξαγωγή στρατιωτικών επιχειρήσεων όσο και στην προστασία των προσωπικών δεδομένων των πολιτών, οδήγησε αναμφίβολα πολλούς ερευνητές, στην προσπάθεια δημιουργίας ψηφιακών υπογραφών στις οποίες η πλαστογράφηση να φαντάζει αδύνατη. Η επίτευξη αυτή, συνάδει με την απόδειξη της μη ύπαρξης λύσεων στα προαναφερόμενα δυσεπίλυτα προβλήματα, η οποία δεν έχει παρατηρηθεί μέχρι σήμερα. Κατ' αυτή την έννοια, οποιοσδήποτε μηχανισμός ψηφιακής υπογραφής μπορεί θεωρητικά να παραβιαστεί πλήρως και ανά πάσα στιγμή.

Στην παρούσα διπλωματική εργασία, αναλύεται το αναγκαίο θεωρητικό υπόβαθρο πάνω στο οποίο στηρίζεται η δομή των εκδοχών και το πρωτόκολλο χρήσης του ECDSA. Περιγράφεται η λειτουργία των συναρτήσεων συμπύκνωσης ενός μηνύματος καθώς και ο καθοριστικός ρόλος των ελλειπτικών καμπυλών πάνω στην ασφάλεια του σχήματος της υπογραφής. Κατόπιν, εισάγονται οι τεχνικές της άλγεβρας των δικτυωτών και περιγράφονται κρυπταναλυτικές επιθέσεις κατά του ECDSA με την χρήση αυτών των τεχνικών, τόσο θεωρητικά όσο και πρακτικά, με την χρήση της γλώσσας προγραμματισμού Python. Ολοκληρώνοντας, αναπτύσσονται ο κώδικας με τον οποίο υπογράφεται και επαληθεύεται ένα μήνυμα και μελετώνται νέες επιθέσεις.

Abstract

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a widely known digital signature scheme which is used by big organizations, including the National Security Agency (NSA) of the USA, in order to certify the origin of the data handled. The functional mechanism of messages' authenticity signed by this scheme, combines intractable historically mathematical problems that besides their aesthetic value, make the term "digital signature" one of the most important achievements of the computer science.

The sensitive issue of information security both in conducting military operations as well as in citizens' personal data protection, led many researchers to create digital signatures hard to forge. This achievement is consistent with the proof of no existing solution to the aforementioned intractable problems, which has not been noticed yet. This way, any digital signature mechanism can theoretically be hacked in anytime.

In this thesis, I analyze the necessary theoretical background, on which the structure of the ECDSA versions and its usage protocol is based on. Furthermore, the density function of a message and the decisive role of the elliptic curves on the signature's key security are described. Moreover, crypto attacks against ECDSA using lattice algebra techniques are described, both theoretically and practically, utilizing the programming language Python. Finally, the code which signs and verifies a message is developed and new attacks are studied.

Ευχαριστίες

Πρώτα απ' όλους, θέλω να ευχαριστήσω τον επιβλέποντα της διπλωματικής μου εργασίας κ. Δημήτριο Πουλάκη Καθηγητή Μαθηματικών Α.Π.Θ, για την πρότασή του να ασχοληθώ με αυτό το καινοτόμο θέμα, την πολύτιμη καθοδήγησή του και τις επεξηγήσεις του. Επίσης, θα ήθελα να ευχαριστήσω την σύζυγό μου Ευανθία, για τη συνεχή στήριξή της σε όλη τη διάρκεια των σπουδών μου.

Κεφάλαιο 1

Κρυπτογραφία και Ψηφιακές Υπογραφές

Σύμφωνα με την υιοθετούμενη μέχρι τώρα εξέλιξη της τεχνολογίας αλλά και του τρόπου επικοινωνίας μεταξύ των ανθρώπων, η ανάγκη διασφάλισης ισχυρότερων μέσων προστασίας της πληροφορίας, επέβαλλαν την ανάπτυξη νέων επιστημονικών τεχνικών που να αντιτάσσονται κάθε φορά στο ύψος των περιστάσεων, σε όλο ένα και μεγαλύτερο βάθος χρόνου. Προς επίτευξη αυτού του σκοπού, εδώ και πολλά χρόνια, έχει στραφεί και η κρυπτογραφία.

Κρυπτογραφία σήμερα, καλείται η μελέτη των μαθηματικών μεθόδων που εξασφαλίζουν την εμπιστευτικότητα, ακεραιότητα, αυθεντικότητα και αδυναμία αποκήρυξης δεδομένων [50]. Η κρυπτογραφία, εκμεταλλεύεται τον ιδιαίτερο ρόλο που χαρακτηρίζει τους επιστημονικούς κλάδους των μαθηματικών και της πληροφορικής και στην κατεύθυνση αυτή, επιστρατεύει νέα πρότυπα και νέες ιδέες. Η μελέτη των μαθηματικών μεθόδων που αποσκοπούν την αναίρεση κάποιων από τις προηγούμενες ιδιότητες, ονομάζεται κρυπτανάλυση [50].

Οι παραπάνω επιστήμες, είχαν πάντοτε μια ιδιαίτερη σημασία για τον στρατό και την διπλωματία. Στην σημερινή όμως εποχή των υπολογιστών, τα πεδία εφαρμογής τους έχουν επεκταθεί σημαντικά. Η ανάγκη προστασίας των ατομικών δικαιωμάτων των πολιτών, η οργάνωση προσωπικών δεδομένων, η ασφάλεια της πληροφορίας στο διαδίκτυο, το οικονομικό σύστημα μια χώρας, είναι μερικά μόνο από αυτά που μπορούν να αναφερθούν.

Στις επόμενες σελίδες, θα αναφερθούμε σε περισσότερους ορισμούς, οι οποίοι είναι απαραίτητοι σε όλη την συνέχεια της εργασίας μας.

1.1 Εισαγωγικές Έννοιες

1.1.1 Βασικοί Ορισμοί

Υποθέτουμε ότι υπάρχουν δύο πρόσωπα Α και Β που επικοινωνούν δια μέσου ενός τυχαίου διαύλου επικοινωνίας. Λέμε ότι η επικοινωνία των Α και Β έχει *ασφαλή μετάδοση της πληροφορίας* αν ικανοποιεί τις εξής ιδιότητες:

1. *Εμπιστευτικότητα:* Κανένας τρίτος να μη μπορεί να λάβει γνώση των μηνυμάτων ή μέρους των που ανταλλάσσονται μεταξύ του Α και του Β.

2. *Ακεραιότητα:* Κανένας τρίτος να μην μπορεί να τροποποιήσει τα μηνύματα που ανταλλάσσονται μεταξύ του Α και του Β.

3. *Αυθεντικότητα:* Καθένας από τους Α και Β πρέπει να είναι σίγουρος για την ταυτότητα του άλλου, την προέλευση των μηνυμάτων, την ημερομηνία τους και τα λοιπά δεδομένα του μηνύματος.

4. *Αδυναμία αποκήρυξης:* Αδυναμία άρνησης των Α και Β της αποστολής ή υπογραφής κάποιου προηγούμενου μηνύματός των.

Ορισμός 1.1. Ένα *σχήμα κρυπτογράφησης* ή *κρυπτοσύστημα* είναι μια πεντάδα συνόλων (P, C, K, E, D) όπου P, C, K καλούνται, αντίστοιχα, *χώρος των απλών κειμένων*, *χώρος των κρυπτογραφημένων κειμένων* και *χώρος των κλειδιών*.

Το E αποτελείται από τις *συναρτήσεις κρυπτογράφησης* $E_k : P \longrightarrow C$ και το D απο τις *συναρτήσεις αποκρυπτογράφησης* $D_k : C \longrightarrow P$, όπου $k \in K$.

Για κάθε $e \in K$ υπάρχει $d \in K$ τέτοια ώστε για κάθε $m \in P$ να ισχύει $D_d(E_e) = m$. Το e καλείται *κλειδί κρυπτογράφησης* και το d *κλειδί αποκρυπτογράφησης* που αντιστοιχεί στο e.

Στη συνέχεια της διπλωματικής, η φράση "για κάθε", θα συμβολίζεται ορισμένες φορές, ποιο απλά με $\forall$.

Ορισμός 1.2. Ένα σχήμα κρυπτογράφησης καλείται *συμμετρικό*, αν $\forall$ $e \in K$ το κλειδί αποκρυπτογράφησης d που αντιστοιχεί στο e είναι δυνατόν να υπολογιστεί πολύ εύκολα από το e, ενώ ένα σχήμα κρυπτογράφησης καλείται *ασύμμετρο* ή *δημοσίου κλειδιού*, αν ο υπολογισμός του d από το e είναι πρακτικά αδύνατος.

Παρατήρηση 1.1. Στην περίπτωση του συμμετρικού κρυπτοσυστήματος, το κλειδί θα πρέπει να κρατείται μυστικό, ενώ στην περίπτωση του ασύμμετρου κρυπτοσυστήματος, το κλειδί κρυπτογράφησης e δημοσιοποιείται και το κλειδί αποκρυπτογράφησης d κρατείται κρυφό.

1.1.2 Τύποι Επίθεσης Κρυπτοσυστήματος

Οι πλέον συνήθεις τύποι επίθεσης κρυπτοσυστήματος, είναι οι εξής:

1. *Επίθεση κρυπτογραφημένου κειμένου:* Το κρυπτογραφημένο κείμενο είναι γνωστό στον κρυπταναλυτή και στόχος είναι η εύρεση του κλειδιού αποκρυπτογράφησης ή αντίστοιχων απλών κειμένων.

2. *Επίθεση γνωστού απλού κειμένου:* Ζεύγη απλών κειμένων με το ίδιο κλειδί και με τα αντίστοιχα κρυπτογραφημένα τους, είναι γνωστά στον κρυπταναλυτή και στόχος είναι η αποκρυπτογράφηση άλλων κειμένων ή εύρεσης του κλειδιού αποκρυπτογράφησης.

3. *Επίθεση επιλεγμένου απλού κειμένου:* Επιλεγμένα απλά κείμενα, χωρίς την γνώση του κλειδιού κρυπτογράφησης, κρυπτογραφούνται από τον κρυπταναλυτή, με στόχο την εύρεση του κλειδιού αποκρυπτογράφησης ή αποκρυπτογράφησης άλλων κειμένων.

4. *Επίθεση επιλεγμένου κρυπτογραφημένου κειμένου:* Επιλεγμένα κρυπτογραφημένα κείμενα, χωρίς την γνώση του κλειδιού αποκρυπτογράφησης, είναι δυνατόν να επιλέγονται από τον κρυπταναλυτή και να αποκρυπτογραφούνται, με στόχο την εύρεση του κλειδιού αποκρυπτογράφησης.

1.1.3 Τέλεια Ασφάλεια

Έστω $A = (P, C, K, E, D)$ ένα κρυπτοσύστημα με αντίστοιχες συναρτήσεις κρυπτογράφησης και αποκρυπτογράφησης E_k και D_k. Θεωρούμε τις κατανομές πιθανότητας P_P και P_K στα σύνολα P και K, αντίστοιχα. Η πιθανότητα να έχουμε ένα απλό κείμενο p το οποίο έχει κρυπτογραφηθεί με ένα κλειδί k, είναι:

$$P(p, k) = P_P(p)P_K(k) \ , \ \forall \ (p, k) \in P \times K.$$

Ας είναι $p \in P$ και $A_p = \{(p, k) : k \in K\}$ ένα ενδεχόμενο, τότε έχουμε:

$$P(A_p) = \sum_{k \in K} P(p, k) = \sum_{k \in K} P_P(p)P_K(k) = P_P(p).$$

Όμοια, για τυχαία επιλογή ενός $k \in K$ και θεωρώντας το ενδεχόμενο $A_k = \{(p,k) : p \in P\}$, έχουμε:

$$P(A_k) = \sum_{p \in P} P(p,k) = \sum_{p \in P} P_P(p) P_K(k) = P_K(k).$$

Συνεπώς καταλήγουμε ότι:

$$P(A_p \cap A_k) = P(p,k) = P_P(p) P_K(k) = P(A_p) P(A_k),$$

σχέση η οποία δείχνει την ανεξαρτησία των ενδεχομένων A_p και A_k.

Ορισμός 1.3. Σύμφωνα με τον C.Shannon, ένα κρυπτοσύστημα διαθέτει *τέλεια ασφάλεια*, αν $\forall\ p \in P$ και $\forall\ c \in C$ ισχύει:

$$P(A_p | A_c) = P(A_p),$$

όπου $A_c = \{(p,k) : E_k(p) = c\}$.

Η προηγούμενη σχέση δείχνει, ότι η γνώση ενός οποιουδήποτε κρυπτογραφημένου κειμένου, δεν γνωστοποιεί στον κρυπταναλυτή το απλό κείμενο από το οποίο προήλθε. Το επόμενο θεώρημα, το οποίο οφείλεται στον C.Shannon (1949) [41], δίνει έναν χαρακτηρισμό των κρυπτοσυστημάτων που διαθέτουν τέλεια ασφάλεια.

Θα συμβολίζουμε με $|S|$, το πλήθος των στοιχείων ενός συνόλου S και θα το ονομάζουμε *τάξη* του συνόλου S.

Θεώρημα 1.1. Έστω $|P| = |K| = |C| < \infty$ και $P(p) > 0$, $\forall\ p \in P$. Ένα κρυπτοσύστημα A έχει τέλεια ασφάλεια, αν και μόνον αν, $\forall\ k \in K$ ισχύει $P(k) = \frac{1}{|K|}$ και $\forall$ ζεύγος $(p,c) \in P \times C$ υπάρχει μοναδικό $k \in K$ τέτοιο ώστε $A_c = \{(p,k)\}$.

Στις μέρες μας, έχει διαπιστωθεί, ότι ο ορισμός που δόθηκε από τον C.Shannon για την τέλεια ασφάλεια δεν αρκεί. Για παράδειγμα, το κρυπτοσύστημα του Vernam, παρόλο που ικανοποιεί το θεώρημα του C.Shannon είναι εξαιρετικά ευάλωτο στην προβολή γνωστού καθαρού κειμένου. Έτσι, είναι απαραίτητο, η ασφάλεια να έχει σταθερότητα, ακόμη και στο ενδεχόμενο της ανάγνωσης κάποιων τμημάτων του κειμένου.

Σύγχρονοι Τύποι Ασφάλειας

Οι επικρατέστεροι τύποι ασφάλειας οι οποίοι μπορούν πρακτικά να αξιολογήσουν ένα κρυπτοσύστημα είναι οι παρακάτω:

1. *Αποδείξιμη ασφάλεια:* Η ασφάλεια αυτή ισοδυναμεί με την δυσκολία επίλυσης ενός γνωστού αριθμο-θεωρητικού προβλήματος, όπως η παραγοντοποίηση μεγάλων ακέραιων αριθμών ή ο διακριτός λογάριθμος.

2. *Υπολογιστική ασφάλεια:* Ακόμη και αν ένας κρυπταναλυτής διαθέτει μια ή περισσότερες από τις ποιο εξελιγμένες τεχνικές κρυπτανάλυσης, καμία από αυτές δεν είναι ικανή να διασπάσει το κρυπτοσύστημα με στοιχειώδη υπολογιστική προσπάθεια. Τα κρυπτοσυστήματα αυτά, στηρίζονται στη δυσκολία επίλυσης συγκεκριμένων προβλημάτων σε εύλογο χρονικό περιθώριο.

3. *Ασφάλεια άνευ όρων:* Ο τύπος αυτής της ασφάλειας, βασίζεται στην υπόθεση οτι: όσο μεγάλος και αν είναι ο όγκος του κρυπτοκειμένου τον οποίο κατέχει ένας κρυπταναλυτής, ακόμη και αν διαθέτει απεριόριστη υπολογιστική ισχύ, δεν υπάρχουν αρκετές πληροφορίες οι οποίες να του δίνουν την δυνατότητα να ανακτήσει το καθαρό κείμενο.

4. *Ασφάλεια θεωρητικής πολυπλοκότητας:* Ο τύπος αυτός της ασφάλειας υφίσταται, όταν ένας ισχυρός κρυπταναλυτής στηρίζεται σε ασθενείς υποθέσεις για την διάσπαση του κρυπτοσυστήματος. Οι υποθέσεις αυτές έχουν να κάνουν με:

   (α′) το μέγεθος των δεδομένων εισόδου της κρυπτανάλυσης,

   (β′) το μέγεθος του χώρου αποθήκευσης της κρυπτανάλυσης,

   (γ′) τον υπολογιστικό χρόνο που απαιτεί η κρυπτανάλυση.

1.2 Ψηφιακή Υπογραφή

Με τον όρο ψηφιακή υπογραφή, εννοούμε ένα μαθηματικό σύστημα, το οποίο χρησιμοποιείται προκειμένου να επαληθευθεί η προέλευση λαμβανόμενων δεδομένων σε ηλεκτρονική μορφή. Η ιδέα της ψηφιακής υπογραφής, ανήκει στους Diffie και Hellman [52], την οποία παρουσίασαν το 1976. Λίγο καιρό αργότερα όμως, οι Rivest, Shamir και Adleman, δημοσίευσαν τον αλγόριθμο RSA, ο οποίος χρησιμοποιήθηκε στις πρώτες ψηφιακές υπογραφές [52].

1.2.1 Διαδικασία Ψηφιακής Υπογραφής

Κάθε φυσικό πρόσωπο, διαθέτει έναν μοναδικό γραφικό χαρακτήρα, μέσο του οποίου, μπορεί και υπογράφει διάφορα έγγραφα, τεκμηριώνοντας με αυτό τον τρόπο, την ταυτότητά του πάνω σε αυτά.

Με την ίδια λογική, η τεκμηρίωση της ταυτότητας ενός προσώπου σε έγγραφα που διατίθενται σε ηλεκτρονική μορφή, πραγματοποιείται με την λεγόμενη ψηφιακή υπογραφή. Στις δύο αυτές περιπτώσεις, οποιοδήποτε τρίτο πρόσωπο, μπορεί εύκολα να συμπεράνει ποιος είναι ο υπογράφων, αν και μόνο αν, αναγνωρίζει επακριβώς, την συγκεκριμένη κάθε φορά υπογραφή. Για να γίνει όμως αυτό, θα πρέπει η διαδικασία της αναγνώρισης, να διέπει την ύπαρξη αυστηρών κριτηρίων, η χρήση των οποίων, να καθιστά ικανή, την διασφάλιση της γνησιότητας και ακεραιότητας της υπογραφής.

Υποθέτουμε ότι ένα πρόσωπο A θέλει να υπογράψει ένα απλό κείμενο p και να το αποστείλει σε ένα πρόσωπο B. Ο A, χρησιμοποιεί ένα μυστικό κλειδί s, αποδίδει την υπογραφή του p με y και στέλνει το ζεύγος (p, y) στον B. Ο B τώρα, για να διαπιστώσει την αυθεντικότητα του μηνύματος, δεν έχει παρά να ελέγξει, αν το y είναι η υπογραφή του p, κάνοντας χρήση ενός δημοσίου κλειδιού v.

Στην καθημερινή μας ζωή, υπάρχουν διάφορες αρχές πιστοποίησης με την μορφή αξιόπιστων οργανισμών, όπου αποθηκεύονται σε βάσεις δεδομένων τα δημόσια κλειδιά. Όλοι αυτοί οι οργανισμοί, διατηρούν διακομιστές με τους οποίους μπορεί κάποιος να επικοινωνεί ηλεκτρονικά και να αντλεί πληροφορίες για δημόσια κλειδιά. Συνεπώς, όταν ένας υπολογιστής ενός χρήστη λαμβάνει μια ψηφιακή υπογραφή, αυτή συνοδεύεται από πληροφορίες, για το ποιά αρχή πιστοποίησης μπορεί να εγγυηθεί για το δημόσιο κλειδί του υπογράφοντα.

Ορισμός 1.4. Λέγοντας *Σχήμα Ψηφιακής Υπογραφής*, αναφερόμαστε σε μια πεντάδα (P, Y, K, S, V), όπου P ο *χώρος των μηνυμάτων*, Y ο *χώρος των υπογραφών*, K ο *χώρος των κλειδιών*, S ο *χώρος των συναρτήσεων υπογραφής* και V ο *χώρος των συναρτήσεων επαλήθευσης*. Τα σύνολα S, V είναι της μορφής:

$$S = \{S_k \ / \ S_k : P \to Y \ , \ k \in K\}$$

και

$$V = \{V_k \ / \ V_k : P \times Y \to \{0, 1\} \ , \ k \in K\} .$$

Για οποιοδήποτε $k \in K$ και $\forall$ ζεύγος $(p, y) \in D_{V_k}$, ισχύει:

$$V_k(p, y) = \begin{cases} 1 & , \ S_k(p) = y \\ 0 & , \ S_k(p) \neq y \end{cases}$$

Το ζεύγος (p, y) ονομάζεται *υπογεγραμμένο μήνυμα*.

Αν p είναι ένα απλό κείμενο και A ένας που υπογράφει το κείμενο αυτό, τότε για κάθε επιλογή $k \in K$ από τον A, η υπογραφή $y = S_k(p)$ γίνεται αποδεκτή από έναν παραλήπτη B, αν και μόνο αν, $V_k(p, y) = 1$.

1.2.2 Τύποι Επίθεσης Ψηφιακής Υπογραφής

Σκοπός της επίθεσης σε μια ψηφιακή υπογραφή είναι η πλαστογράφησή της. Ανάλογα με το είδος της επίθεσης, η πλαστογράφηση εμπίπτει σε μια ή περισσότερες από τις παρακάτω περιπτώσεις:

1. *Υπαρξιακή πλαστογράφηση (existential forgery):* Ο επιτιθέμενος έχει την δυνατότητα να υπογράψει έγκυρα, ένα τουλάχιστον κείμενο, αλλά η ενέργειά του μπορεί να γίνει αντιληπτή από τον αποστολέα.

2. *Επιλεκτική πλαστογράφηση (selective forgery):* Ο επιτιθέμενος γνωρίζει τις έγκυρες υπογραφές συγκεκριμένων κειμένων ή κάποιας κατηγορίας αυτών και υπογράφει κατά περίπτωση, ενεργώντας έμμεσα ως αποστολέας.

3. *Ολική πλαστογράφηση (universal forgery):* Ο επιτιθέμενος δεν γνωρίζει το μυστικό κλειδί του αποστολέα, άλλα έχει προσδιορίσει μια διαδικασία ισοδύναμη με αυτή του αποστολέα, η οποία του παρέχει έγκυρες υπογραφές.

4. *Ολική Κατάρρευση (total break):* Ο επιτιθέμενος έχει καταφέρει να υπολογίσει το μυστικό κλειδί, το οποίο χρησιμοποιεί ο αποστολέας για την υπογραφή των κειμένων του.

Το σύνολο των επιθέσεων στην ψηφιακή υπογραφή, χωρίζεται στις δύο επόμενες κατηγορίες:

1. *Επιθέσεις κλειδιού (key-only attacks):* Ο επιτιθέμενος γνωρίζει μόνο το δημόσιο κλειδί του αποστολέα.

2. *Επιθέσεις μηνύματος (message attacks):* Ο επιτιθέμενος γνωρίζει τις υπογραφές ενός συνόλου μηνυμάτων.

   (α′) *Επίθεση γνωστού μηνύματος (known-message attacks):* Στον επιτιθέμενο είναι γνωστές οι υπογραφές μιας αυθαίρετης λίστας μηνυμάτων, την οποία δεν έχει επιλέξει ο ίδιος.

(β΄) *Επίθεση επιλεγμένου μήνυματος (chosen-message attack):* Στον επιτιθέμενο είναι γνωστές οι υπογραφές μιας συγκεκριμένης λίστας μηνυμάτων, την οποία έχει επιλέξει πριν υλοποιηθούν οι υπογραφές.

(γ΄) *Επίθεση προσαρμόσιμου επιλεγμένου μηνύματος (adaptive chosen-message attack):* Ο επιτιθέμενος ζητάει έγκυρες υπογραφές μηνυμάτων που εξαρτώνται από το δημόσιο κλειδί του αποστολέα ή σχετίζονται με προηγούμενες υπογραφές ή μηνύματα.

Οι επιθέσεις που υλοποιούνται σε αυτή την διπλωματική εργασία, ανήκουν στις κατηγορία των επιθέσεων μηνύματος (message attacks), με σκοπό την ολική κατάρρευση (total break) του συστήματος της ψηφιακής υπογραφής.

Κεφάλαιο 2

Ακέραιοι Αριθμοί και Αλγεβρικές Δομές

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε θεμελιώδεις έννοιες από την θεωρία αριθμών και την αφηρημένη άλγεβρα, το φάσμα των οποίων βρίσκει ευρεία εφαρμογή στην κρυπτογραφία. Για περισσότερες πληροφορίες, ο αναγνώστης μπορεί να ανατρέξει στα συγγράμματα [49], [51].

2.1 Στοιχεία Θεωρίας Αριθμών

Τα στοιχειώδη αντικείμενα μελέτης της θεωρίας αριθμών είναι οι ιδιότητες των συνόλων:

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

και

$$\mathbb{N} = \{0, 1, 2, \ldots\},$$

τις οποίες θα αναπτύξουμε στην ενότητα αυτή. Οι αριθμοί του συνόλου $\mathbb{Z}$ ονομάζονται *ακέραιοι* και οι αριθμοί του συνόλου $\mathbb{N}$ ονομάζονται *φυσικοί*.

2.1.1 Διαιρετότητα

Ας είναι $a, b, c, d \in \mathbb{Z}$.

Ορισμός 2.1. Θα λέμε οτι ο ακέραιος $a \neq 0$ *διαιρεί* τον ακέραιο b, αν και μόνο αν, $b = ca$, για κάποιον ακέραιο c.

Με τον συμβολισμό $a \mid b$, θα εκφράζουμε την διαίρεση του b από τον a, ενώ με $a \nmid b$ την περίπτωση που ο b δεν διαιρείται από τον a. Ισχύει επομένως η παρακάτω ισοδυναμία:

$$a \mid b \iff b = ca \ , \text{ για κάποιο } c \in \mathbb{Z}.$$

Παράδειγμα 2.1. $3 \mid 27,\ 2 \mid 8,\ 7 \mid 21$ όμως $7 \nmid 22$.

Η μελέτη της στοιχειώδους θεωρίας αριθμών, μας δίνει ορισμένα κριτήρια διαιρετότητας, τα οποία ενισχύουν την ταχύτητα των υπολογισμών μας, στην προσπάθεια να βρούμε τους διαιρέτες ενός ακέραιου. Τα κριτήρια αυτά είναι ιδιαίτερα χρήσιμα όταν ο ακέραιος για τον οποίο αναζητούμε τους διαιρέτες του είναι αρκετά μεγάλος.

Πρόταση 2.1. Ας είναι x ένας ακέραιος. Τότε:

1. $10 \mid x$ αν το τελευταίο ψηφίο του x είναι 0.
2. $2 \mid x$ αν το τελευταίο ψηφίο του x είναι 0, 2, 4, 6 ή 8.
3. $5 \mid x$ αν το τελευταίο ψηφίο του x είναι 0 ή 5.
4. $3 \mid x$ ή $9 \mid x$ αν το άθροισμα των ψηφίων του x διαιρείται με το 3 ή το 9, αντίστοιχα.
5. $4 \mid x$ και $25 \mid x$ αν τα δύο τελευταία ψηφία του x είναι 0.
6. Έστω b ο ακέραιος που αποτελείται από τα k τελευταία ψηφία του ακέραιου x και a ο ακέραιος που αποτελείται από τα l πρώτα ψηφία του ακέραιου x και $k+l$ να είναι το πλήθος των ψηφίων του ακέραιου x. Αν ισχύει $a \mid b$ τότε ισχύει και $a \mid x$.

Για το τελευταίο κριτήριο, δίνεται το επόμενο παράδειγμα:

Παράδειγμα 2.2. Έστω ο ακέραιος

$$x = 24578245893827488912619760571140096001594816686801932635254 7.$$

Αφού παρατηρήσουμε ότι

$$x = \underbrace{24578245893827488912619760571}_{a}\underbrace{140096001594816686801932635254 7}_{b}$$

με

$$140096001594816686801932635254 7 = 57 \cdot 24578245893827488912619760571,$$

τότε και $a \mid x$.

Σημείωση: $x = a \cdot (10000000000000000000000000000000 + 57)$.

Πρόταση 2.2. Μερικές βασικές ιδιότητες της διαιρετότητας είναι οι παρακάτω:

1. $a \mid b$ και $b \neq 0 \ \Rightarrow \ |a| \leq |b|$.
2. $a \mid b$ και $b \mid a \ \Rightarrow \ |a| = |b|$.
3. $a \mid b \ \Rightarrow \ ax \mid bx \ , \ \forall \ x \in \mathbb{Z} - \{0\}$.
4. $a \mid b$ και $b \mid c \ \Rightarrow \ a \mid c$.
5. $a \mid b$ και $c \mid d \ \Rightarrow \ ac \mid bd$.
6. $a \mid b$ και $a \mid c \ \Rightarrow \ a \mid bx + cy \ , \ \forall \ x, y \in \mathbb{Z}$.
7. $a \mid bc \Rightarrow a \mid b$ ή $a \mid c$.

Θεώρημα 2.1. Για μοναδικό ζεύγος $(q, r) \in \mathbb{Z}^2$ ισχύει:

$$b = aq + r \ , \ 0 \leq r < |a| \, .$$

Η σχέση αυτή, ονομάζεται *Ευκλείδεια διαίρεση*.

Πόρισμα 2.1. Για μοναδικό ζεύγος $(q, r) \in \mathbb{Z}^2$ ισχύει:

$$b = 2q + r \ , \ 0 \leq r < 2$$

Αν $r = 0$ ο ακέραιος b λέγεται *άρτιος*, ενώ αν $r = 1$ ο ακέραιος b λέγεται *περιττός*.

Ας είναι

$$A = \{a_i \in \mathbb{Z} : i = 1, 2, \ldots, n\}$$

για τυχαίο φυσικό αριθμό $n > 1$ και

$$D_A = \{d \in \mathbb{Z} : \ d \mid a \ , \ \forall \ a \in A\} \, .$$

Συμβολίζουμε με $\min S$ το μικρότερο στοιχείο ενός συνόλου S και $\max S$ το μεγαλύτερο στοιχείο ενός συνόλου S.

Για το σύνολο D_A ισχύει:

$$D_A \neq \oslash \text{ και } |D_A| < \infty,$$

αφού

$$1 \in D_A \text{ και } |D_A| \leq \min \{|a_i| : a_i \in A\} \, ,$$

αντίστοιχα.

Ορισμός 2.2. Κάθε στοιχείο d_i $(i = 1, 2, \ldots, |D_A|)$ του συνόλου D_A, ονομάζεται *κοινός διαιρέτης* των στοιχείων του A. Αν $d = \max\{d_i : d_i \in D_A\}$, τότε ο d ονομάζεται *μέγιστος κοινός διαιρέτης* των στοιχείων του A.

Θα συμβολίζουμε με $\mu\kappa\delta(A)$ τον μέγιστο κοινό διαιρέτη των στοιχείων του συνόλου A ή απλά με $\mu\kappa\delta(a_1, a_2, \ldots, a_n)$ τον μέγιστο κοινό διαιρέτη των στοιχείων a_i, $i = 1, 2, \ldots, n$.

Οι a, b ονομάζονται *πρώτοι μεταξύ τους*, αν και μόνο αν, $\mu\kappa\delta(a, b) = 1$

Θεώρημα 2.2. Αν $d = \mu\kappa\delta(A)$, τότε θα υπάρχουν ακέραιοι k_i, $i = 1, 2, \ldots, n$ (όχι μοναδικοί), έτσι ώστε

$$d = k_1a_1 + k_2a_2 + \cdots + k_na_n.$$

Πόρισμα 2.2. Έστω ένας θετικός ακέραιος $d \in D_A$, τότε θα ισχύει η παρακάτω ισοδυναμία:

$$d = \mu\kappa\delta(A) \Longleftrightarrow c \mid d \ , \ \forall\ c \in D_A$$

όπου c θετικός ακέραιος.

Πόρισμα 2.3. Αν $a \mid bc$ και $\mu\kappa\delta(a, b) = 1$ τότε $a \mid c$.

Λήμμα 2.1. Αν r είναι το υπόλοιπο της διαίρεσης του b από τον a, τότε ισχύει:

$$\mu\kappa\delta(a, b) = \mu\kappa\delta(b, r).$$

2.1.2 Πρώτοι Αριθμοί

Ένα από τα βασικότερα αντικείμενα μελέτης της θεωρίας αριθμών είναι οι πρώτοι αριθμοί.

Ορισμός 2.3. Έστω ένας θετικός ακέραιος p. Ο p ονομάζεται *πρώτος*, αν δεν διαιρείται από άλλον ακέραιο εκτός από το 1 και το p. Αν ο p δεν είναι πρώτος, τότε ονομάζεται *σύνθετος*.

Θεώρημα 2.3. Το σύνολο των πρώτων αριθμών είναι άπειρο και συμβολίζεται με $\mathbb{P}$.

Έστω τυχαίος θετικός ακέραιος $a > 1$.

Ορισμός 2.4. Ονομάζουμε *πρώτο διαιρέτη* του a, κάθε διαιρέτη του a που είναι πρώτος.

Θεώρημα 2.4. Ο a αναλύεται κατά μοναδικό τρόπο στο γινόμενο

$$p_1^{k_1} p_2^{k_2} \cdots p_n^{k_n}$$

όπου $k_i \geq 1$ θετικοί ακέραιοι και $p_i \in \mathbb{P}$, $i = 1, 2, \ldots, n$.

Η παραπάνω μορφή ονομάζεται *πρωτογενής ανάλυση* του a.

Η εύρεση της πρωτογενούς ανάλυσης ενός θετικού ακέραιου ονομάζεται *παραγοντοποίηση*. Όταν ο ακέραιος αυτός είναι σχετικά μικρός, η παραγοντοποίησή του πραγματοποιείται εύκολα, σε αντίθετη περίπτωση, εφαρμόζονται ειδικές μέθοδοι. Εδώ, αξίζει να επισημάνουμε, ότι οι ακέραιοι a και $10000000000000000000000000000057$ του Παραδείγματος 2.2 είναι πρώτοι.

2.1.3 Σχέσεις Ισοτιμίας

Ορισμός 2.5. Έστω n ένας θετικός ακέραιος και $a, b \in \mathbb{Z}$. Ο a λέμε οτι είναι *ισότιμος* με τον b *κατά μέτρο* n και το συμβολίζουμε με $a \equiv b \bmod n$, αν $n \mid a - b$.

Παράδειγμα 2.3. Έχουμε οτι $23 \equiv 3 \bmod 5$ και $231748 \equiv 0 \bmod 23$, αφού $5 \mid 23 - 3$ και $23 \mid 231748$, αντίστοιχα.

Πρόταση 2.3. Έστω $a, b, c, d \in \mathbb{Z}$. Τότε ισχύουν τα παρακάτω:

1. Αν $a \equiv b \bmod n$ και $b \equiv c \bmod n$, τότε $a \equiv c \bmod n$.
2. Αν $a \equiv b \bmod n$ και $c \equiv d \bmod n$ τότε θα είναι $a + c \equiv b + d \bmod n$ και $ac \equiv bd \bmod n$.
3. Αν $a \equiv b \bmod n$ τότε θα είναι $ka \equiv kb \bmod n$ και $a^k \equiv b^k \bmod n$ για κάθε θετικό ακέραιο k.
4. Αν p είναι πρώτος, τότε $(a + b)^p \equiv a^p + b^p \bmod p$.

Ορισμός 2.6. Έστω $a, n \in \mathbb{Z}$ με $n > 1$. Ονομάζουμε *κλάση ισοτιμίας* του a και το συμβολίζουμε με $[a]_n$, το σύνολο

$$[a]_n = \{x \in \mathbb{Z} : x \equiv a \bmod n\} .$$

Ορισμός 2.7. Συμβολίζουμε με $\varphi(n)$ το πλήθος των φυσικών που είναι μικρότεροι και πρώτοι με τον n. Η συνάρτηση

$$\varphi : \mathbb{N} \to \mathbb{N} - \{0\} \ , \ n \mapsto \varphi(n)$$

καλείται *συνάρτηση του Euler*.

Παράδειγμα 2.4. Παρατηρούμε ότι $\varphi(6) = 2$ και $\varphi(7) = 6$.

Θεώρημα 2.5. Για κάθε $m, n \in \mathbb{N} - \{0\}$ με $\mu\kappa\delta\,(m, n) = 1$ ισχύει:

$$\varphi(mn) = \varphi(m)\varphi(n).$$

Πόρισμα 2.4. Έστω $n \in \mathbb{N} - \{0\}$ με πρωτογενή ανάλυση $p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$. Τότε ισχύει:

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right) \cdots \left(1 - \frac{1}{p_m}\right).$$

Θεώρημα 2.6. Έστω $a, n \in \mathbb{Z}$ με $n > 1$ και $\mu\kappa\delta\,(a, n) = 1$. Τότε ισχύει:

$$a^{\varphi(n)} \equiv 1 \bmod n.$$

Το παραπάνω θεώρημα είναι γνωστό ως *Θεώρημα των Fermat-Euler*.

Παράδειγμα 2.5. Επιλέγοντας $a = 5$ και $n = 6$ έχουμε $\mu\kappa\delta\,(5, 6) = 1$ και $\varphi(6) = 2$. Επομένως θα είναι:

$$5^2 \equiv 1 \bmod 6$$

Πόρισμα 2.5. Έστω $p \in \mathbb{P}$ και a ένας ακέραιος με $p \nmid a$. Τότε ισχύει:

$$a^{p-1} \equiv 1 \bmod p.$$

Το παραπάνω αποτέλεσμα είναι γνωστό ως *μικρό Θεώρημα του Fermat*.

Θεώρημα 2.7. Έστω $a, b, x, n \in \mathbb{Z}$ με $n > 1$ και $\mu\kappa\delta\,(a, n) = 1$. Τότε ισχύει:

$$ax \equiv b \bmod n \iff x \equiv ba^{\varphi(n)-1} \bmod n.$$

Ορισμός 2.8. Έστω $a, n \in \mathbb{Z}$ με $n > 1$ και $\mu\kappa\delta\,(a, n) = 1$. Ο μικρότερος θετικός ακέραιος r για τον οποίο ισχύει $a^r \equiv 1 \bmod n$, καλείται *τάξη του ακέραιου a κατά μέτρο n* και συμβολίζεται με $ord_n(a)$.

Αν $r = ord_n(a)$ τότε για οποιονδήποτε ακέραιο $k > r$, ισχύει η ισοδυναμία: $a^k \equiv 1 \bmod n \Leftrightarrow r \mid k$. Στην περίπτωση που $r = \varphi(n)$, ο a καλείται *αρχική ή πρωτογενής ρίζα κατά μέτρο n*.

Ορισμός 2.9. Έστω $a, n \in \mathbb{Z}$ με $n > 1$ και $\mu\kappa\delta\,(a, n) = 1$. Ο a ονομάζεται *υπόλοιπο $k-$οστής δύναμης κατά μέτρο n*, αν η πολυωνυμική ισοτιμία

$$x^k \equiv a \bmod n$$

έχει λύση. Στην περίπτωση που $k = 2$, ο a ονομάζεται *τετραγωνικό υπόλοιπο κατά μέτρο n* και οι λύσεις της πολυωνυμικής ισοτιμίας, ονομάζονται *τετραγωνικές ρίζες του a κατά μέτρο n*.

Στην επόμενη πρόταση, παρουσιάζεται μια ειδική περίπτωση εύρεσης τετραγωνικών ριζών, με μέτρο έναν πρώτο.

Πρόταση 2.4. Έστω $p \in \mathbb{P}$ με $p \equiv 3 \bmod 4$ και a ένα τετραγωνικό υπόλοιπο κατά μέτρο p. Τότε, οι λύσεις της τετραγωνικής ισοτιμίας

$$x^2 \equiv a \bmod p$$

είναι

$$x \equiv \pm a^{(p+1)/4} \bmod p. \tag{2.1}$$

2.1.4 Παράσταση Ακεραίων και Πράξεις

Ένα αρκετά σημαντικό αποτέλεσμα του Θεωρήματος 2.1 είναι και το επόμενο θεώρημα:

Θεώρημα 2.8. Έστω ένας θετικός ακέραιος $g > 1$. Οποιοσδήποτε θετικός ακέραιος a αναλύεται κατά μοναδικό τρόπο στο άθροισμα

$$\sum_{i=1}^{k} a_i g^{k-i} = a_1 g^{k-1} + \cdots + a_k \tag{2.2}$$

όπου $k = \lfloor \log_g a \rfloor + 1$ και

$$a_i = \left\lfloor (a - \sum_{j=1}^{i-1} a_j g^{k-j})/g^{k-i} \right\rfloor \in \{0, \ldots, g-1\}$$

με $i = 1, \ldots, k$ και $a_1 \neq 0$.

Με $\lfloor x \rfloor$, συμβολίζουμε τον μεγαλύτερο ακέραιο, που είναι μικρότερος ή ίσος του x ενώ με $\lceil x \rceil$, συμβολίζουμε τον μικρότερο ακέραιο, που είναι μεγαλύτερος ή ίσος του x.

Ορισμός 2.10. Η μορφή (2.2) του Θεωρήματος 2.8, ονομάζεται *$g-$αδική παράσταση (ή μορφή) του ακέραιου a* και συμβολίζεται με

$$a = (a_1 \dots a_k)_g \text{ ή ποιο απλά με } a = a_1 \dots a_k.$$

Στην περίπτωση που $g > 10$, αν υπάρχουν ακέραιοι $a_i \in \{10, \dots, g-1\}$ για κάποια $i \in \{1, \dots, k\}$, τότε αυτοί αναπαρίστανται διαδοχικά, με γράμματα από το λατινικό αλφάβητο. Πχ: αν $g = 16$, τότε τα γράμματα a, b, c, d, e, f (πεζά ή κεφαλαία) ισοδυναμούν με τους αριθμούς 10, 11, 12, 13, 14, 15, αντίστοιχα.

Ορισμός 2.11. Το πλήθος των ψηφίων της $g-$ αδικής παράστασης ενός ακέραιου a, καλείται *μήκος του a στην $g-$αδική παράσταση* και συμβολίζεται με $l_g(a)$.

Συνήθως, ως μήκος του ακέραιου a, εννοούμε τον αριθμό $l_2(a)$.

Παρατήρηση 2.1. Σύμφωνα με το Θεώρημα 2.8 και τον Ορισμό 2.11,

$$\text{αν } g^k > a \geq g^{k-1} \text{ τότε } k = l_g(a)$$

και αντίστροφα.

Τα ψηφία 0 και 1 αποτελούν την βασική δομή λειτουργίας των ηλεκτρονικών υπολογιστών με την οποία αναπαριστούν και επεξεργάζονται δεδομένα. Τα ψηφία αυτά ονομάζονται *δυαδικά ψηφία* ή *bits (binary digits)*.

Ορισμός 2.12. Έστω $x = (x_1 \dots x_k)_2$ και $y = (y_1 \dots y_l)_2$ δύο ακέραιοι και $z = (z_1 \dots z_s)_2 = x * y$, όπου $* \in \{+, -, \cdot, \div\}$. Κάθε διαδικασία η οποία χρησιμοποιείται για τον υπολογισμό οποιουδήποτε δυαδικού ψηφίου z_i , $i = 1, \dots, s$ καλείται *δυαδική ψηφιακή πράξη* [51].

Πρόταση 2.5. Έστω x και y δύο ακέραιοι όπου $y = xq + r$ η Ευκλείδεια διαίρεσή τους. Αν k, l είναι τα μήκη τους αντίστοιχα, τότε ο αριθμός:

1. $x \pm y$ προκύπτει με $\max\{k, l\}$ δυαδικές ψηφιακές πράξεις.
2. $x \cdot y$ προκύπτει με το πολύ $(k-1)l$ δυαδικές ψηφιακές πράξεις.

3. q και r προκύπτουν με το πολύ $k(l-k+1)$ δυαδικές ψηφιακές πράξεις.

Ιδιότητα 2.1. Κάθε δυνατός συνδυασμός από 4 ακριβώς bits, αναπαριστά έναν θετικό ακέραιο μικρότερο του 16 και αντίστροφα, ένας θετικός ακέραιος μικρότερος του 16, παρίσταται από έναν συνδυασμό με 4 ακριβώς bits.

Σύμφωνα με την Ιδιότητα 2.1, η επόμενη πρόταση μας δείχνει πως η $2-$αδική παράσταση ενός θετικού ακέραιου, μπορεί εύκολα να μετατραπεί στην $16-$αδική του παράσταση και αντίστροφα.

Πρόταση 2.6. Έστω x ένας θετικός ακέραιος. Αν $x=(a_1\dots a_q)_2$, $q=l_2(x)$ και $z=4\left\lceil\frac{q}{4}\right\rceil-q$ τότε $x=(b_1\dots b_r)_{16}$, $r=l_{16}(x)$ όπου

$$\begin{aligned} b_1 &= \underbrace{0\dots0}_{z\ \text{φορές}}a_1\dots a_{4-z}\\ b_2 &= a_{5-z}\dots a_{8-z}\\ \vdots &\ \ \vdots\ \ \vdots\\ b_r &= a_{r-3}\dots a_r \end{aligned}$$

Αντίστροφα, αν $x=(b_1\dots b_r)_{16}$, $r=l_{16}(x)$, $r_1=l_2(b_1)$ και $q=4(r-1)+r_1$ τότε $x=(a_1\dots a_q)_2$ όπου

$$\begin{aligned} b_1 &= \underbrace{0\dots0}_{4-r_1\ \text{φορές}}\ a_1\dots a_{r_1}\\ b_2 &= a_{r_1+1}\dots a_{r_1+4}\\ \vdots &\ \ \vdots\ \ \vdots\\ b_r &= a_{q-3}\dots a_q \end{aligned}$$

Παράδειγμα 2.6. Έστω οι αριθμοί $x=(10101001111010011000110100 1011)_2$ και $y=(e2bc03d5)_{16}$. Για τον αριθμό x έχουμε

$$x=\overbrace{\underbrace{00}_{2}\ 10}^{\text{επέκταση}}\ \underbrace{1010}_{a}\ \underbrace{0111}_{7}\ \underbrace{1010}_{a}\ \underbrace{0110}_{6}\ \underbrace{0011}_{3}\ \underbrace{0100}_{4}\ \underbrace{1011}_{b}$$

επομένως $x = (2a7a634b)_{16}$ ενώ για τον αριθμό y έχουμε

$$\begin{aligned} e &= 1110 \\ 2 &= 0010 \\ b &= 1011 \\ c &= 1100 \\ 0 &= 0000 \\ 3 &= 0011 \\ d &= 1101 \\ 5 &= 0101 \end{aligned}$$

επομένως $y = (11100010101111000000001111010101)_2$.

Στο επόμενο παράδειγμα, προστίθενται δύο θετικοί ακέραιοι, οι οποίοι είναι στην $16-$αδική τους μορφή και στη συνέχεια δίνεται το υπόλοιπο της διαίρεσης του αθροίσματος με 16^5.

Παράδειγμα 2.7. Έστω $x = (ec2b9)_{16}$ και $y = (7fd3b)_{16}$ τότε έχουμε

$$\begin{aligned} x + y &= 14 \cdot 16^4 + 12 \cdot 16^3 + 2 \cdot 16^2 + 11 \cdot 16^1 + 9 \cdot 16^0 \\ &+ 7 \cdot 16^4 + 15 \cdot 16^3 + 13 \cdot 16^2 + 3 \cdot 16^1 + 11 \cdot 16^0 \\ &= 21 \cdot 16^4 + 27 \cdot 16^3 + 15 \cdot 16^2 + 14 \cdot 16^1 + 20 \cdot 16^0 \\ &= 21 \cdot 16^4 + 27 \cdot 16^3 + 15 \cdot 16^2 + 14 \cdot 16^1 + (16+4)16^0 \\ &= 21 \cdot 16^4 + 27 \cdot 16^3 + 15 \cdot 16^2 + (1+14)16^1 + 4 \cdot 16^0 \\ &= 21 \cdot 16^4 + (16+11)16^3 + 15 \cdot 16^2 + 15 \cdot 16^1 + 4 \cdot 16^0 \\ &= (16+6)16^4 + 11 \cdot 16^3 + 15 \cdot 16^2 + 15 \cdot 16^1 + 4 \cdot 16^0 \\ &= 16^5 + 6 \cdot 16^4 + 11 \cdot 16^3 + 15 \cdot 16^2 + 15 \cdot 16^1 + 4 \cdot 16^0 \end{aligned}$$

επομένως $(x + y) \equiv (6bff4)_{16} \bmod 16^5$.

Γενικότερα, για τον υπολογισμό του υπολοίπου ενός οποιουδήποτε θετικού ακέραιου με έναν ακέραιο της μορφής g^m, μπορούμε να χρησιμοποιήσουμε την ακόλουθη πρόταση:

Πρόταση 2.7. Έστω $a = (a_1 \dots a_k)_g$ ένας θετικός ακέραιος στην $g-$αδική του μορφή και m ένας φυσικός με $1 \leq m < k$. Τότε, το υπόλοιπο του αριθμού αυτού με τον ακέραιο g^m είναι ο ακέραιος $\upsilon = (a_{k-m+1} \dots a_k)_g$.

Σύμφωνα με την Πρόταση 2.7 αν a είναι ένας θετικός ακέραιος και θέλουμε να υπολογίσουμε το υπόλοιπό του με έναν ακέραιο της μορφής g^m όπου $a > g^m$ και $m \in \{1\} \cup \mathbb{P}$, αρκεί να υπολογίσουμε την $g-$αδική παράσταση του ακέραιου a. Ομοίως, αν d είναι ένας διαιρέτης του m τότε μπορούμε να υπολογίσουμε την g^d-αδική παράσταση του a.

2.2 Αλγεβρικές Δομές

2.2.1 Ομάδες

Ορισμός 2.13. Ένα σύνολο G εφοδιασμένο με μία διμελή πράξη

$$* : G \times G \to G \ , \ (x, y) \mapsto x * y$$

ονομάζεται *ομάδα* και συμβολίζεται με $(G, *)$, αν και μόνο αν, $\forall\, x, y, z \in G$ ικανοποιούνται οι παρακάτω ιδιότητες:

1. Η πράξη $*$ είναι προσεταιριστική: $x * (y * z) = (x * y) * z$.
2. Υπάρχει ένα στοιχείο $e \in G$, το οποίο ονομάζεται *ουδέτερο*, τέτοιο ώστε: $e * x = x * e = x$.
3. Υπάρχει στοιχείο $x' \in G$, το οποίο ονομάζεται *συμμετρικό* του x, τέτοιο ώστε $x' * x = x * x' = e$.

Αν η πράξη $*$ είναι και αντιμεταθετική, δηλαδή αν $x * y = y * x$ για κάθε $x, y \in G$, τότε η ομάδα ονομάζεται *αβελιανή*.

Παράδειγμα 2.8. Τα σύνολα $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$ των ακέραιων, ρητών, πραγματικών και μιγαδικών αριθμών αντίστοιχα, εφοδιασμένα με την πράξη $+$, αποτελούν αβελιανές ομάδες.

Ορισμός 2.14. Έστω $(G, *)$ μια ομάδα και ένα μη κενό σύνολο $F \subseteq G$. Το σύνολο F ονομάζεται *υποομάδα* της G και συμβολίζεται με $F \leqslant G$ ή $(F, *) \leq (G, *)$, αν και μόνο άν, ικανοποιούνται οι παρακάτω ιδιότητες:

1. Για κάθε $x, y \in F \Rightarrow x * y \in F$.
2. Για κάθε $x \in F \Rightarrow x' \in F$.

Σε περίπτωση που έχουμε $F \subset G$, η F καλείται *γνήσια* υποομάδα της G και συμβολίζεται με $F < G$.

Παράδειγμα 2.9. Παρατηρούμε ότι $(\mathbb{Z}, +) < (\mathbb{Q}, +) < (\mathbb{R}, +) < (\mathbb{C}, +)$.

Θεώρημα 2.9. Έστω $(F, *) \leq (G, *)$ με $|G| < \infty$. Τότε, η τάξη της υποομάδας $(F, *)$ διαιρεί την τάξη της ομάδας $(G, *)$. Το θεώρημα αυτό είναι γνωστό ως *Θεώρημα του Lagrange*.

Θεώρημα 2.10. Έστω G μια ομάδα και a ένα στοιχείο της. Το σύνολο

$$F = \{a^n : n \in \mathbb{Z}\}$$

είναι μια υποομάδα της G και μάλιστα με την μικρότερη τάξη που περιέχει το a. Σε αυτήν την περίπτωση, λέμε ότι το στοιχείο a *παράγει* την υποομάδα F και συμβολίζεται με $F =< a >$.

Ορισμός 2.15. Μια ομάδα G ονομάζεται *κυκλική*, αν υπάρχει $a \in G$ τέτοιο ώστε $G =< a >$. Στην περίπτωση αυτή, το στοιχείο a ονομάζεται *γεννήτορας* της G.

Ορισμός 2.16. Ονομάζεται *τάξη* ενός στοιχείου a μια ομάδας G και συμβολίζεται με $ord\,(a)$, ο πληθικός αριθμός $| < a > |$.

Σύμφωνα με το θεώρημα του Lagrange, η τάξη του στοιχείου a διαιρεί την τάξη της ομάδας G.

Παράδειγμα 2.10. Η ομάδα $(\mathbb{Z}, +)$ είναι κυκλική, αφού

$$\mathbb{Z} =< 1 >=< -1 > .$$

Επομένως η τάξη του 1 και του -1 είναι άπειρη. Αντίθετα η τάξη του 0 είναι 1 διότι $< 0 >= \{0\}$.

Πρόταση 2.8. Έστω $G = \{1, a, \ldots, a^{n-1}\}$ μία πεπερασμένη κυκλική ομάδα τάξης n. Ένα οποιοδήποτε στοιχείο $a^m \in G$ είναι γεννήτορας της G, αν και μόνο αν, ισχύει $\mu\kappa\delta\,(m, n) = 1$. Δηλαδή:

$$G =< a^m > \Leftrightarrow \ \mu\kappa\delta\,(m, n) = 1.$$

2.2.2 Δακτύλιοι-Σώματα

Ορισμός 2.17. Ένα σύνολο G εφοδιασμένο με δύο διμελείς πράξεις

$$\oplus : G \times G \to G \ , \ (x, y) \mapsto x \oplus y$$

$$\odot : G \times G \to G \ , \ (x, y) \mapsto x \odot y$$

ονομάζεται *δακτύλιος* και συμβολίζεται με $(G, \oplus, \odot)$, αν και μόνο αν, $\forall\ x, y, z \in G$ ικανοποιούνται οι παρακάτω ιδιότητες:

1. Το ζεύγος $(G, \oplus)$ είναι αβελιανή ομάδα.

2. Η πράξη $\odot$ είναι προσεταιριστική: $x \odot (y \odot z) = (x \odot y) \odot z$.

3. Η πράξη $\odot$ είναι επιμεριστική ώς προς την $\oplus$, δηλαδή $x \odot (y \oplus z) = (x \odot y) \oplus (x \odot z)$ και $(x \oplus y) \odot z = (x \odot z) \oplus (y \odot z)$.

Αν επιπλέον η πράξη $\odot$ είναι αντιμεταθετική, ο δακτύλιος ονομάζεται *αντιμεταθετικός*, ενώ αν υπάρχει ουδέτερο στοιχείο της πράξης αυτής, συμβολίζεται με **1** και ο δακτύλιος ονομάζεται *δακτύλιος με μονάδα*. Το ουδέτερο στοιχείο της ομάδας $(G, \oplus)$ συμβολίζεται με **0** και ονομάζεται *μηδενικό*.

Τα συμμετρικά στοιχεία της ομάδας $(G, \oplus)$ ονομάζονται *αντίθετα* ενώ τα συμμετρικά στοιχεία (αν υπάρχουν) ως προς την πράξη $\odot$ ονομάζονται *αντίστροφα*.Το σύνολο όλων των στοιχείων του G που έχουν αντίστροφο, συμβολίζεται με G^*.

Παράδειγμα 2.11. Τα σύνολα $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ εφοδιασμένα με τις συνήθεις πράξεις της πρόσθεσης και του πολλαπλασιασμού είναι αντιμεταθετικοί δακτύλιοι με μονάδα και $\mathbb{Q}^* = \mathbb{Q} - \{0\}$, $\mathbb{R}^* = \mathbb{R} - \{0\}$ και $\mathbb{C}^* = \mathbb{C} - \{0\}$.

Έστω a, b δύο ακέραιοι με $b > 0$. Θα συμβολίζουμε με $a \bmod b$ το υπόλοιπο της διαίρεσης του a με το b.

Πρόταση 2.9. Το σύνολο $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ που έχει ως στοιχεία όλα τα καθαρά υπόλοιπα της διαίρεσης ενός ακεραίου με n και πράξεις

$$\oplus : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n \ , \ (x, y) \mapsto (x + y) \bmod n$$

$$\odot : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n \ , \ (x, y) \mapsto x \cdot y \bmod n$$

αποτελεί αντιμεταθετικό δακτύλιο με μονάδα το 1.

Ορισμός 2.18. Έστω $(G, \oplus, \odot)$ ένας αντιμεταθετικός δακτύλιος με μονάδα. Αν κάθε μη μηδενικό στοιχείο του δακτυλίου έχει αντίστροφο, τότε ο δακτύλιος αυτός ονομάζεται *σώμα*.

Για να είναι ο δακτύλιος $\mathbb{Z}_n$ σώμα, θα πρέπει $\mathbb{Z}_n = \{0\} \cup \mathbb{Z}_n^*$ ή αλλιώς θα πρέπει $\forall\ x \in \mathbb{Z}_n - \{0\}$ να υπάρχει $y \in \mathbb{Z}_n - \{0\}$ τέτοιο ώστε $xy \equiv 1 \bmod n$.

Παράδειγμα 2.12. Έστω οι αντιμεταθετικοί δακτύλιοι με μονάδα: $\mathbb{Z}$, $\mathbb{Z}_6$ και $\mathbb{Z}_7$. Έχουμε $\mathbb{Z}^* = \{-1, 1\}$, $\mathbb{Z}_6^* = \{1, 5\}$ και $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$. Επομένως μόνον ο δακτύλιος $\mathbb{Z}_7$ είναι σώμα, αφού $\mathbb{Z}_7 = \{0\} \cup \mathbb{Z}_7^*$.

Πρόταση 2.10. $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : \mu\kappa\delta\,(x, n) = 1\}$.

Πόρισμα 2.6. $|\mathbb{Z}_n^*| = \varphi(n)$.

Πόρισμα 2.7. Ένας δακτύλιος $\mathbb{Z}_n$ αποτελεί σώμα, αν και μόνο αν, ο αριθμός n είναι πρώτος.

Παρατήρηση 2.2. Έστω $\mathbb{Z}_n$ ένα σώμα. Τότε από το Θεώρημα $2.1.2$ θα ισχύει:

$$\forall\ x \in \mathbb{Z}_n - \{0\} \Rightarrow x^{-1} \equiv x^{\varphi(n)-1} \bmod n$$

Ορισμός 2.19. Έστω $(G, \oplus, \odot)$ ένας δακτύλιος. Αν υπάρχει ελάχιστος θετικός ακέραιος c τέτοιος ώστε

$$cx = 0\ ,\ \forall\ x \in G$$

τότε λέμε ότι ο δακτύλιος αυτός έχει *χαρακτηριστική* c και θα το συμβολίζουμε με $char(G) = c$. Αν ο δακτύλιος $(G, \oplus, \odot)$ έχει μονάδα το 1, τότε $char(G) = c$, αν και μόνο αν, ο c είναι ο ελάχιστος θετικός αριθμός τέτοιος ώστε $c1 = 0$.

2.2.3 Αρχικές ρίζες κατά μέτρο p

Στην παράγραφο αυτή, θα εξηγήσουμε πότε ένα στοιχείο που ανήκει στο πεπερασμένο σώμα $\mathbb{Z}_p^*$, αποτελεί μια αρχική ρίζα κατά μέτρο p, όπου $p \in \mathbb{P}$. Στη συνέχεια, θα περιγράψουμε την διαδικασία εύρεσης μιας αρχικής ρίζας κατά μέτρο p η οποία ανήκει στο πεπερασμένο σώμα $\mathbb{Z}_p$.

Θεώρημα 2.11. Έστω το πεπερασμένο σώμα $\mathbb{Z}_p$ όπου $p \in \mathbb{P}$. Τότε, για κάθε διαιρέτη d του $p - 1$, υπάρχουν ακριβώς $\varphi(d)$ στοιχεία με τάξη d στο $\mathbb{Z}_p^*$.

Πόρισμα 2.8. Έστω $p \in \mathbb{P}$. Τότε, υπάρχουν ακριβώς $\varphi(p - 1)$ στοιχεία του $\mathbb{Z}_p^*$ που είναι αρχικές ρίζες κατά μέτρο p.

Απόδειξη. Σύμφωνα με το Θεώρημα 2.11, $p-1 \mid p-1$ και συνεπώς υπάρχουν ακριβώς $\varphi(p-1)$ στοιχεία με τάξη $p-1$ στο $\mathbb{Z}_p^*$. Από τον ορισμό όμως, $p-1=\varphi(p)$ και έτσι τα $\varphi(p-1)$ στοιχεία είναι όλες οι αρχικές ρίζες κατά μέτρο p του $\mathbb{Z}_p^*$. □

Θεώρημα 2.12. Έστω $p \in \mathbb{P}$ και $x \in \mathbb{Z}_p^*$. Αν $p_1^{k_1} p_2^{k_2} \cdots p_n^{k_n}$ είναι η πρωτογενής ανάλυση του $p-1$ και $\gamma^{\frac{p-1}{p_i}} \not\equiv 1 \bmod p$ για κάθε $i=1,\ldots,n$, τότε ο γ είναι μια αρχική ρίζα κατά μέτρο p.

Η διαδικασία εύρεσης μιας αρχικής ρίζας κατά μέτρο p στο πεπερασμένο σώμα $\mathbb{Z}_p$, είναι η ακόλουθη:

1. Υπολογίζεται η πρωτογενής ανάλυση $q_1^{k_1} q_2^{k_2} \cdots q_n^{k_n}$ του $p-1$.
2. Επιλέγεται τυχαία ένας ακέραιος $\gamma \in \{2,\ldots,p-2\}$.
3. Αν για κάθε $i=1,\ldots,n$ ισχύει $\gamma^{\frac{p-1}{q_i}} \not\equiv 1 \bmod p$, τότε ο γ είναι μία αρχική ρίζα κατά μέτρο p, αλλιώς επιλέγουμε διαφορετικό ακέραιο γ και επαναλαμβάνουμε την διαδικασία.

Με την παραπάνω διαδικασία, είναι δυνατόν, να βρεθούν όλες οι αρχικές ρίζες κατά μέτρο p στο $\mathbb{Z}_p$.

Η πρόταση που ακολουθεί στη συνέχεια, είναι αρκετά σημαντική, καθώς εξηγεί την θεωρητική βάση δημιουργίας των κλειδιών της ψηφιακής υπογραφής DSA, που θα δούμε αναλυτικά στο Κεφάλαιο 6.

Πρόταση 2.11. Ας είναι $g=\gamma^{\frac{p-1}{q}} \bmod p$ ένα στοιχείο του $\mathbb{Z}_p^*$, όπου p,q πρώτοι αριθμοί, $q \mid p-1$ και γ μια αρχική ρίζα κατά μέτρο p στο $\mathbb{Z}_p^*$. Θα αποδείξουμε ότι το σύνολο $G=\{1,g,\ldots,g^{q-1}\}$ είναι η μοναδική κυκλική υποομάδα με τάξη q του $\mathbb{Z}_p^*$ και περιέχει ακριβώς $q-1$ στοιχεία με τάξη q.

Απόδειξη. Το στοιχείο g έχει τάξη q. Πράγματι, $g^q \equiv \gamma^{p-1} \equiv 1 \bmod p$, ενώ από τον Ορισμό 2.8, έχουμε $q=ord_p(g)$, διαφορετικά θα υπήρχε αριθμός $r=ord_p(g)>1$ τέτοιος ώστε $r \mid q$, το οποίο είναι άτοπο.

Το στοιχείο g παράγει την κυκλική υποομάδα $G=\{1,g,\ldots,g^{q-1}\}$ τάξης q του $\mathbb{Z}_p^*$. Αν $i=1,\ldots,q-1$ τότε για τις τάξεις k_i των g^i ισχύει $\mu\kappa\delta\,(k_i,q)=1$ και από την Πρόταση 2.8 θα έχουμε $G=<g^i> \Rightarrow\ |G|=|<g^i>|=q$.

Επειδή ισχύει $q \mid p-1$ και από το Θεώρημα 2.11, το σώμα $\mathbb{Z}_p^*$ θα περιέχει ακριβώς $\varphi(q)=q-1$ στοιχεία με τάξη q. Επομένως όλα αυτά

τα στοιχεία ανήκουν στην G, η οποία θα είναι η μοναδική κυκλική υποομάδα με τάξη q του $\mathbb{Z}_p^*$. □

Κεφάλαιο 3

Αλγόριθμοι

3.1 Στοιχεία Θεωρίας Αλγορίθμων

Ένας *αλγόριθμος*, είναι μια απόλυτα ακριβής διεργασία από συγκεκριμένα, πεπερασμένου πλήθους βήματα, τα οποία οδηγούν στην λύση ενός προβλήματος. Με την σχεδίαση αλγορίθμων ασχολήθηκαν πρώτοι οι αρχαίοι Έλληνες και οι Κινέζοι Μαθηματικοί, ενώ σήμερα αποτελούν την καρδιά της επιστήμης των υπολογιστών.

Στη σημερινή εποχή των υπολογιστών, η κρυπτογράφηση και η αποκρυπτογράφηση μεγάλου όγκου δεδομένων είναι δυνατόν να πραγματοποιηθεί, ακόμη και σε ελάχιστο χρόνο, με σχετικά μεγάλη ευκολία. Ωστόσο, υπάρχουν πολλά και διαφορετικά κρυπτοσυστήματα με πολυσύνθετη λειτουργική δομή, η διαχείριση των οποίων, απαιτεί την γνώση και εφαρμογή, όλο ένα και περισσότερων παραμέτρων.

Για την επίλυση τέτοιων προβλημάτων στην κρυπτογραφία, έχουν αναπτυχθεί αρκετοί αποτελεσματικοί αλγόριθμοι, οι οποίοι ονομάζονται *κρυπταλγόριθμοι*. Οι κρυπταλγόριθμοι είναι εκτελέσιμοι, τόσο με την χρήση ηλεκτρονικού υπολογιστή, όσο και χωρίς αυτόν. Η επιλογή ενός συγκεκριμένου κάθε φορά κρυπταλγόριθμου, για την επίλυση ενός προβλήματος στην κρυπτογραφία, που επιλύεται και από άλλους κρυπταλγόριθμους, ανατίθεται στην κρίση του κρυπτογράφου ή κρυπταναλυτή, ο οποίος θα εξετάσει την απόδοσή τους σε σχέση με τις ανάγκες του.

3.1.1 Αποδοτικότητα και Χρόνος Εκτέλεσης

Ένα εύλογο ερώτημα που τίθεται εδώ, είναι το εξής: *Τι καλούμε αποδοτικότητα σε έναν αλγόριθμο και πώς την μετράμε;*

Η παραπάνω έννοια είναι ζωτικής σημασίας για έναν αλγόριθμο. Αρκεί να φανταστεί κανείς, ότι υπάρχει ένας αρκετά σημαντικός ερευνητικός τομέας, που ασχολείται αποκλειστικά, με την αξιολόγηση αποδοτικών αλγορίθμων, ικανών να επιλύουν, υπολογιστικά δυσπρόσιτα μαθηματικά προβλήματα.

Εξετάζοντας λοιπόν, μια περισσότερο μαθηματική άποψη της παραπάνω έννοιας, αναφερόμαστε στην ικανότητα των αλγορίθμων, να είναι ευέλικτοι στην επίλυση δυσπρόσιτων μαθηματικών προβλημάτων ανεξαρτήτου μεγέθους, μέσα από σύντομες, ευφυείς και γρήγορες μεθόδους. Κατά κύριο λόγο επομένως, η αποδοτικότητα είναι συνυφασμένη με τον χρόνο εκτέλεσης ενός αλγορίθμου.

Εννοώντας χρόνο εκτέλεσης, δεν αναφερόμαστε αποκλειστικά στην πραγματική έννοια του χρόνου λειτουργίας ενός αλγορίθμου από την είσοδο των δεδομένων σε αυτόν, μέχρι την έξοδο των αποτελεσμάτων από αυτόν. Ο χρόνος εκτέλεσης είναι μια πολύπλοκη έννοια, η οποία συνάδει με το υπολογιστικό κόστος κάθε στοιχειώδους βήματος του αλγορίθμου. Στην πράξη, είναι ο αριθμός όλων των στοιχειωδών υπολογιστικών βημάτων κάθε πεπερασμένης εσωτερικής λειτουργίας του αλγορίθμου, σε συνάρτηση με το μέγεθος των δεδομένων της εισόδου του. Συνήθως, αναφέρεται με τον όρο *χρονική πολυπλοκότητα*.

Η χρονική πολυπλοκότητα ενός αλγορίθμου $\mathcal{A}$ συμβολίζεται με $T(\mathcal{A})$ και είναι μια συνάρτηση της μορφής:

$$T : \mathbb{N}^m \to \mathbb{N} \ , \ (k_1, \ldots, k_m) \longmapsto T\,(k_1, \ldots, k_m)$$

η οποία εκφράζει το πλήθος των δυαδικών ψηφιακών πράξεων που απαιτούνται για την εκτέλεσή του. Οι μεταβλητές k_i, $i = 1, \ldots, m$ είναι τα μήκη (σε bits) m το πλήθος δεδομένων εισόδου $a_i \in \mathbb{Z}$, $i = 1, \ldots, m$, αντίστοιχα.

Παράδειγμα 3.1. Θεωρούμε έναν αλγόριθμο $\mathcal{A}$, ο οποίος υπολογίζει την $n-$ οστή δύναμη ενός ακέραιου a, όπως παρακάτω:

Είσοδος: Θετικοί ακέραιοι a, n.
Έξοδος: $s = a^n$.

1. $s \leftarrow a$
2. Για $i = 1$ μέχρι $n - 1$

   (α΄) $s \leftarrow s \cdot a$

3. Επίστρεψε s

Παρατηρούμε ότι η εντολή 2.(α') θα εκτελεστεί $n-1$ φορές, επομένως αρκεί να υπολογίσουμε τις δυαδικές ψηφιακές πράξεις που απαιτούνται σε κάθε επανάληψη και να τις αθροίσουμε. Η χρονική πολυπλοκότητα αυτού του αλγορίθμου, υπολογίζεται από τον επόμενο πίνακα:

Επανάληψη	Μέγιστος αριθμός δυαδικών ψηφιακών πράξεων
1^{η}	$(l_2(a)-1)l_2(a)$
2^{η}	$(l_2(a)-1)l_2(a^2)$
$\vdots$	$\vdots$
$(n-1)^{\eta}$	$(l_2(a)-1)l_2(a^{n-1})$

Έχουμε:

$$\begin{aligned} T(a,n) &\leq (l_2(a)-1)\left(l_2(a)+l_2(a^2)+\cdots+l_2(a^{n-1})\right) \\ &< l_2(a)\left(l_2(a)+2l_2(a)+\cdots+(n-1)l_2(a)\right) \\ &= \tfrac{(n-1)n}{2}l_2^2(a) \end{aligned}$$

όπου

$$\begin{aligned} l_2(a^k) &= \left\lfloor \log_2 a^k \right\rfloor + 1 = k\left\lfloor \log_2 a \right\rfloor + 1 \\ &< k\left(\left\lfloor \log_2 a \right\rfloor + 1\right) = kl_2(a) \end{aligned}$$

για οποιοδήποτε $k \geq 1$ και

$$1+2+\cdots+n-1 = \frac{(n-1)n}{2}.$$

Συνεπώς, η χρονική πολυπλοκότητα αυτού του αλγορίθμου είναι το πολύ, το ανωτέρω άνω φράγμα που υπολογίστηκε.

Ορισμός 3.1. Λέμε ότι ένας αλγόριθμος είναι *πολυωνυμικού χρόνου*, αν ο διπλασιασμός των μηκών k_i, $i=1,\ldots,m$ των δεδομένων της εισόδου του, αυξάνει την χρονική πολυπλοκότητα $T(k_1,\ldots,k_m)$, κατά ένα σταθερό θετικό παράγοντα.

Ένας αλγόριθμος θεωρείται *αποδοτικός*, αν είναι πολυωνυμικού χρόνου.

3.1.2 Ασυμπτωτική Πολυπλοκότητα

Ο προσδιορισμός της χρονικής πολυπλοκότητας ενός αλγορίθμου, δεν μπορεί πάντοτε να είναι ακριβής. Παραδείγματος χάριν, έστω ότι ένας αλγόριθμος είναι σχεδιασμένος να εκτελεί αναζήτηση ονομάτων σε έναν

κατάλογο με n αταξινόμητα ονόματα. Προφανώς, ο αλγόριθμος αυτός θα σταματήσει την αναζήτηση, όταν βρει ένα επιθυμητό όνομα, διαφορετικά θα συνεχίσει την αναζήτηση με τον έλεγχο όλων των n ονομάτων του καταλόγου.

Η διαδικασία προσδιορισμού της χρονικής πολυπλοκότητας αυτού του αλγορίθμου, είναι παρόμοια με αυτή του Παραδείγματος 3.1 και υπολογίζεται ο χρόνος στην χειρότερη περίπτωση που ένα όνομα είναι ανύπαρκτο. Ο χρόνος αυτός αποτελεί στην ουσία ένα άνω φράγμα για οποιαδήποτε είσοδο ονόματος και καλείται *χρόνος χειρότερης περίπτωσης*.

Η σύγκριση της χρονικής πολυπλοκότητας ενός ή περισσότερων αλγορίθμων μεταξύ τους, δεν είναι εύκολη υπόθεση. Για παράδειγμα, μπορεί να υπάρχει ένας αλγόριθμος $\mathcal{A}_1$, ο οποίος μετά την είσοδο δεδομένων μήκους d_1, να τελειώνει σε χρόνο $T(\mathcal{A}_1)$, ενώ ένας αλγόριθμος $\mathcal{A}_2$, μετά την είσοδο δεδομένων μήκους $d_2 > d_1$, να τελειώνει σε χρόνο $T(\mathcal{A}_2) < T(\mathcal{A}_1)$. Ωστόσο, ενδέχεται η "συμπεριφορά" του $\mathcal{A}_1$ καθώς τα μήκη d_1 λαμβάνουν μεγαλύτερες τιμές, να προσεγγίσει, ακόμη και να αλλάξει σε αυτήν του $\mathcal{A}_2$. Αν αυτό συμβεί, τότε έχουμε μια σαφή ένδειξη, ότι ο χρόνος του $\mathcal{A}_1$ βελτιώνεται συγκριτικά με αυτόν του $\mathcal{A}_2$, εντός ενός συγκεκριμένου διαστήματος τιμών μήκους. Με άλλα λόγια, ο ρυθμός με τον οποίο αυξάνεται ο χρόνος του $\mathcal{A}_1$, ενδεχομένως να είναι μικρότερος από αυτόν του $\mathcal{A}_2$ για απεριόριστα μήκη δεδομένων.

Συγκρίνοντας την χρονική πολυπλοκότητα ενός ή περισσότερων αλγορίθμων μεταξύ τους, έχει ιδιαίτερη σημασία να εξετάσουμε την ικανότητά τους να επεξεργάζονται όλο ένα και μεγαλύτερο όγκο δεδομένων, χωρίς να αυξάνεται σημαντικά ο χρόνος τους. Αυτό που μας ενδιαφέρει είναι η συμπεριφορά των αλγορίθμων καθώς το μέγεθος των δεδομένων της εισόδου λαμβάνει απεριόριστες τιμές. Συγκρίνουμε με άλλα λόγια, τον ρυθμό αύξησης του χρόνου χειρότερης περίπτωσης ενός αλγορίθμου με τον ρυθμό αύξησης του χρόνου χειρότερης περίπτωσης ενός άλλου αλγορίθμου, καθώς το μέγεθος των δεδομένων εισόδου $a_i \to \infty$, $i = 1, \ldots, m$. Ο ρυθμός αυτός ονομάζεται *ασυμπτωτικός ρυθμός αύξησης*.

Ας είναι $T(\mathcal{A})$ η χρονική πολυπλοκότητα ενός αλγορίθμου $\mathcal{A}$. Η συνάρτηση T λέμε ότι έχει *ασυμπτωτικό ρυθμό αύξησης με άνω φράγμα* $f(k_1, \ldots, k_m)$, αν υπάρχουν σταθερές $c > 0$ και $s_i \geq 0$, $i = 1, \ldots, m$ τέτοιες ώστε:

$$\forall\ k_i \geq s_i\ (i = 1, \ldots, m)\ \Rightarrow\ T(k_1, \ldots, k_m) \leq cf(k_1, \ldots, k_m).$$

Στην περίπτωση αυτή λέμε, ότι η *ασυμπτωτική πολυπλοκότητα* του αλ-

γορίθμου $\mathcal{A}$ είναι $O(f(k_1, \ldots, k_m))$, το οποίο συμβολίζεται με:

$$T(\mathcal{A}) = O(f(k_1, \ldots, k_m)).$$

Παράδειγμα 3.2. Έστω ένας αλγόριθμος $\mathcal{A}$ ο οποίος δέχεται ως είσοδο ένα δεδομένο ($m = 1$). Αν $T(k) = 2k^3 + 7k + 5$ τότε $T(\mathcal{A}) = O(k^3)$. Πράγματι: $2k^3 + 7k + 5 < 2k^3 + 7k^3 + 5k^3 = 14k^3$. Επομένως αρκεί να επιλέξουμε $c = 14$ και $s = 1$.

Παράδειγμα 3.3. Η ασυμπτωτική πολυπλοκότητα του αλγορίθμου του Παραδείγματος 3.1 είναι $O(k^2 \log_2^2 k)$ όπου $k = \max\{a, n\}$. Πράγματι:

$$\begin{aligned} T(a,n) &< n^2 l_2^2(a) < n^2 (\log_2 a + 1)^2 < n^2 (2 \log_2 a)^2 \\ &= 4n^2 \log_2^2 a < 4 (\max\{a,n\})^2 \log_2^2 (\max\{a,n\}) \\ &= 4k^2 \log_2^2 k. \end{aligned}$$

Επομένως αρκεί να επιλέξουμε $c = 4$ και $s = 2$.

Στη συνέχεια της διπλωματικής, όταν θα αναφερόμαστε στην χρονική πολυπλοκότητα ενός αλγορίθμου, θα εννοούμε την ασυμπτωτική πολυπλοκότητα με τον συμβολισμό $O(f(k))$.

Πρόταση 3.1. Οι συνηθέστερες χρονικές πολυπλοκότητες των αλγορίθμων, είναι οι παρακάτω:

1. *Σταθερή:* $O(1)$.
2. *Λογαριθμική:* $O(\log k)$.
3. *Γραμμική:* $O(k)$.
4. *Τετραγωνική:* $O(k^2)$.
5. *Κυβική:* $O(k^3)$.
6. *Πολυωνυμική:* $O(k^d)$, $d > 0$.
7. *Εκθετική:* $O(d^k)$, $d > 1$.

Στον επόμενο πίνακα δίνονται ενδεικτικά, οι πραγματικοί χρόνοι εκτέλεσης αλγορίθμων με διάφορες χρονικές πολυπλοκότητες και για αυξανόμενο μέγεθος, ενός δεδομένου εισόδου k.

	$O(k)$	$O(k \log k)$	$O(k^2)$	$O(k^3)$	$O\left(2^k\right)$
$k = 10$	1 sec	1 sec	1 sec	1 sec	1 sec
$k = 100$	1 sec	1 sec	1 sec	1 sec	10^{17} έτη
$k = 10.000$	1 sec	1 sec	2 λεπτά	12 ημέρες	$> 10^{25}$ έτη
$k = 1.000.000$	1 sec	20 έτη	12 ημέρες	31.710 έτη	$> 10^{25}$ έτη

3.2 Σχεδίαση και Εφαρμογές Αλγορίθμων

3.2.1 Αλγόριθμοι και Υπολογιστική Θεωρία Αριθμών

Στην ενότητα αυτή, παρουσιάζουμε ορισμένους χρήσιμους αλγόριθμους, με αντίστοιχα παραδείγματα, οι οποίοι εφαρμόζονται για την επίλυση υπολογιστικών προβλημάτων της θεωρίας αριθμών.

Έστω a,b δύο ακέραιοι αριθμοί. Με βάση το Θεώρημα 2.1, ο υπολογισμός του $\mu\kappa\delta(a,b)$ πραγματοποιείται εύκολα, με τον επόμενο *αλγόριθμο του Ευκλείδη*:

$$\begin{array}{rcll} b & = & aq + r\ , & 0 \leq r < a \\ a & = & rq_1 + r_1\ , & 0 \leq r_1 < r \\ r & = & r_1q_2 + r_2\ , & 0 \leq r_2 < r_1 \\ \vdots & & & \vdots \\ r_{n-1} & = & r_nq_{n+1} + r_{n+1}\ , & 0 \leq r_{n+1} < r_n \\ r_n & = & r_{n+1}q_{n+2} + r_{n+2}\ , & 0 \leq r_{n+2} < r_{n+1} \end{array}$$

όπου $0 \leq r_{n+2} < r_{n+1} < \cdots < r < a$.

Ο παραπάνω αλγόριθμος τερματίζει, όταν κάποιο από τα υπόλοιπα λάβει τιμή μηδέν, καθώς σε διαφορετική περίπτωση, θα είχαμε άπειρους ακέραιους αριθμούς στο διάστημα $(0,a)$, το οποίο είναι αδύνατον. Εδώ υποθέτουμε ότι $r_{n+2} = 0$, επομένως σύμφωνα με το Λήμμα 2.1, έχουμε:

$$\mu\kappa\delta(a,b) = \mu\kappa\delta(r,a) = \mu\kappa\delta(r_1,r) = \cdots = \mu\kappa\delta(r_{n+1},r_n) = r_{n+1}.$$

Η χρονική πολυπλοκότητα αυτού του αλγόριθμου είναι $O(l_2(a)l_2(b))$.

Παράδειγμα 3.4. Θα προσδιορίσουμε τον $\mu\kappa\delta\,(691,73)$. Έχουμε:

$$\begin{array}{rcl} 691 & = & 73 \cdot 9 + 34 \\ 73 & = & 34 \cdot 2 + 5 \\ 34 & = & 5 \cdot 6 + 4 \\ 5 & = & 4 \cdot 1 + 1 \\ 4 & = & 1 \cdot 4 + 0 \end{array}$$

επομένως $\mu\kappa\delta\,(691,73) = 1$.

Από το Θεώρημα 2.8 προκύπτει ο επόμενος αλγόριθμος που υπολογίζει την $g-$αδική παράσταση ενός ακεραίου.

Αλγόριθμος 3.1. Υπολογισμός $g-$αδικής παράστασης ακεραίου.

Είσοδος: Θετικός ακέραιος a.
Έξοδος: $g-$αδική παράσταση του a.

1. $k \leftarrow \lfloor \log_g a \rfloor + 1$.
2. $A_1 \leftarrow a$.
3. Για $i = 1$ μέχρι k με βήμα 1

   (α′) $a_i \leftarrow \lfloor A_i / g^{k-i} \rfloor$,
   (β′) $A_{i+1} \leftarrow A_i - a_i g^{k-i}$.
4. Επίστρεψε $a_1 \dots a_k$.

Η χρονική πολυπλοκότητα αυτού του αλγόριθμου είναι $O(l_g^2(a))$.

Παράδειγμα 3.5. Έστω ο ακέραιος 51. Θα υπολογίσουμε την $2-$αδική του παράσταση, αφού πρώτα προσδιορίσουμε το μήκος του.

Έχουμε $64 = 2^6 > 51 > 2^5 = 32$, επομένως $l_2(51) = 6$ και τα δυαδικά του ψηφία $a_i,\ i = 1, \dots, 6$ είναι:

$$\begin{aligned} a_1 &= \lfloor 51/2^5 \rfloor = 1 \\ a_2 &= \lfloor (51 - 2^5)/2^4 \rfloor = 1 \\ a_3 &= \lfloor (51 - 2^5 - 2^4)/2^3 \rfloor = 0 \\ a_4 &= \lfloor (51 - 2^5 - 2^4)/2^2 \rfloor = 0 \\ a_5 &= \lfloor (51 - 2^5 - 2^4)/2^1 \rfloor = 1 \\ a_6 &= \lfloor (51 - 2^5 - 2^4 - 2^1)/2^0 \rfloor = 1 \end{aligned}$$

Συνεπώς $51 = 2^5 + 2^4 + 2^1 + 2^0 = (110011)_2$ ενώ σε $16-$αδική μορφή $51 = 3 \cdot 16^1 + 3 \cdot 16^0 = (33)_{16}$.

Ο επόμενος αλγόριθμος, υπολογίζει το υπόλοιπο του αθροίσματος δύο θετικών ακεραίων x, y που είναι στην $g-$αδική τους μορφή, με g^m, όπου $1 \leq m \leq k$ και $k = \max\{l_g(x), l_g(y)\}$, χωρίς να χρειαστεί να υπολογίσουμε ολόκληρη την $g-$αδική παράσταση του αθροίσματος $x+y$. Για $g = 2$ και $m = k = 32$ η έξοδος αυτού του αλγόριθμου είναι η πράξη (5.5) που θα δούμε στο Κεφάλαιο 5.

Αλγόριθμος 3.2. Πρόσθεση ακεραίων mod g^m.

Είσοδος: Θετικοί ακέραιοι $x = (x_1 \dots x_q)_g\ ,\ y = (y_1 \dots y_r)_g$ και m.
Έξοδος: Άθροισμα $z = (x + y) \bmod g^m$.

1. $k \leftarrow \max\{q, r\}$
2. Αν $q \neq r$ τότε
   (α′) Αν $k = q$ τότε $y_1 \dots y_k \leftarrow \underbrace{0 \dots 0}_{k-r \text{ φορές}}\ y_1 \dots y_r$.
   (β′) Αν $k = r$ τότε $x_1 \dots x_k \leftarrow \underbrace{0 \dots 0}_{k-q \text{ φορές}}\ x_1 \dots x_q$.
3. $S_k \leftarrow x_k + y_k$
4. $a_k \leftarrow S_k \bmod g$
5. Για $i = 1$ μέχρι $m - 1$ με βήμα 1
   (α′) $S_{k-i} \leftarrow x_{k-i} + y_{k-i} + \frac{S_{k+1-i} - a_{k+1-i}}{g}$,
   (β′) $a_{k-i} \leftarrow S_{k-i} \bmod g$.
6. $z \leftarrow a_{k-m+1} \dots a_k$.
7. Επίστρεψε z.

Η χρονική πολυπλοκότητα αυτού του αλγόριθμου είναι $O(k^3)$.

Παράδειγμα 3.6. Το υπόλοιπο του αθροίσματος του Παραδείγματος 2.7, εκτελώντας τον αλγόριθμο 3.2 για $m = 5$, προκύπτει από τον επόμενο πίνακα:

n	$=$	5	4	3	2	1
S_n	$=$	20	15	15	27	22
a_n	$=$	4	f	f	b	6

Για $m = 1$ από τον ίδιο πίνακα, έχουμε $(x + y) \bmod 16 = 4$.

Ο επόμενος αλγόριθμος, υπολογίζει το υπόλοιπο της ύψωσης ενός ακεραίου a σε κάποια δύναμη k, με n, όπου $0 \leq a \leq n$ και k στην $2-$αδική του μορφή.

Αλγόριθμος 3.3. Ύψωση σε δύναμη $\bmod n$.

Είσοδος: Θετικοί ακέραιοι a και $k = (k_1 \dots k_r)_2$.
Έξοδος: Ύψωση σε δύναμη $z = a^k \bmod n$.

1. $A_1 \leftarrow a$

2. $P_1 \leftarrow a^{k_r}$

3. Για $i = 2$ μέχρι r με βήμα 1

   (α′) $A_i \leftarrow A_{i-1}^2 \bmod n$

   (β′) $P_i \leftarrow A_i^{k_{r-i}} P_{i-1} \bmod n$

4. $z \leftarrow P_r$

5. Επίστρεψε z.

Η χρονική πολυπλοκότητα αυτού του αλγόριθμου είναι $O(rl_2^2(n))$.

Παράδειγμα 3.7. Ο υπολογισμός του ακέραιου $z = 24^{37} \bmod 149$, όπου $37 = (100101)_2$ εκτελώντας τον αλγόριθμο 3.2, προκύπτει από τον επόμενο πίνακα:

i	$=$	1	2	3	4	5	6
A_i	$=$	24	129	102	123	80	142
P_i	$=$	24	24	64	64	64	148

Συνεπώς $z = P_6 = 148$.

3.2.2 Το Πρόβλημα του Διακριτού Λογάριθμου

Έστω $(G, *)$ μία ομάδα και g ένα στοιχείο της με τάξη n. Όπως ήδη αναφέραμε, η κυκλική υποομάδα της G που παράγεται από το g είναι η $< g >$. Αν $a \in < g >$ τότε υπάρχει μοναδικός ακέραιος x με $0 \leq x \leq n-1$ τέτοιος, ώστε $g^x = a$.

Ορισμός 3.1. Το πρόβλημα της εύρεσης του ακέραιου x όταν είναι γνωστά τα στοιχεία g και a, ονομάζεται *Πρόβλημα του Διακριτού Λογάριθμου (Discrete Logarithm Problem-DLP)*. Ο ακέραιος x, συμβολίζεται με $\log_g a$ και καλείται *διακριτός λογάριθμος* του a ως προς την βάση g.

Η απλούστερη μέθοδος με την οποία υπολογίζεται ο διακριτός λογάριθμος είναι η μέθοδος της απαρίθμησης, δηλαδή, ο υπολογισμός των δυνάμεων $g^1, g^2, g^3, \ldots$, με τον αλγόριθμο 3.3 έως ότου βρεθεί ένα x για το οποίο $g^x = a$.

Παράδειγμα 3.1. Έχουμε:

$$7^x \equiv 176 \bmod 2017 \text{ με } x = 1088$$

Επομένως $\log_7 176 = 1088$. Παρατηρούμε ότι ο υπολογισμός του ακέραιου x με την απλή μέθοδο της απαρίθμησης, χρειάζεται περίπου 1087 υπολογισμούς $(\bmod\ 2017)$.

Η απλή μέθοδος της απαρίθμησης, δεν είναι καθόλου αποτελεσματική όταν χρησιμοποιείται σε κρυπτογραφικές εφαρμογές που απαιτούν την εύρεση του διακριτού λογάριθμου. Οι περισσότερες από αυτές τις εφαρμογές, στην περίπτωση της ομάδας $\mathbb{Z}_p^*$, απαιτούν η τιμή του ακέραιου x, να είναι τουλάχιστον 2^{160}. Η εύρεση του διακριτού λογάριθμου για πολλές ομάδες, όπως οι υποομάδες της $\mathbb{Z}_p^*$, θεωρείται δύσκολο πρόβλημα και μέχρι σήμερα δεν είναι γνωστός κάποιος αλγόριθμος πολυωνυμικού χρόνου που να το επιλύει.

Η χρησιμότητα του προβλήματος του διακριτού λογάριθμου, από την άποψη της κρυπτογραφίας, έγκειται, στο συνδυασμό της δυσκολίας του προβλήματος και της ευκολίας του υπολογισμού μίας δύναμης $(\bmod\ p)$. Ας σημειωθεί, ότι για να διατηρεί το πρόβλημα του διακριτού λογάριθμου την δυσκολία του και κατά συνέπεια την χρησιμότητά του στην κρυπτογραφία, πρέπει ο πρώτος p να έχει μήκος μεγαλύτερο από 768 και επίσης, να μην είναι κάποιας ειδικής μορφής για την οποία οι υπάρχοντες αλγόριθμοι είναι αποτελεσματικοί.

Μερικοί από τους κυριότερους αλγόριθμους που επιλύουν τον DLP όχι όμως σε πολυωνυμικό χρόνο, είναι οι παρακάτω:

1. Αλγόριθμος Shanks (ή Baby step/Giant step).
2. ρ - Αλγόριθμος Pollard.
3. Αλγόριθμος των Pohlig-Hellman.
4. Αλγόριθμος Λογισμού – Δεικτών (Index Calculus).

Για τους προαναφερόμενους αλγόριθμους, ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει στο σύγγραμμα [2].

Κεφάλαιο 4

Ελλειπτικές Καμπύλες

4.1 Θεωρία Ελλειπτικών Καμπυλών

Οι ελλειπτικές καμπύλες έχουν τις ρίζες τους στην δουλειά του *Διόφαντου* $(250$ μχ$)$ και στους *Abel* και *Jacobi*. Από τότε η θεωρία τους έχει μελετηθεί εκτενώς για σχεδόν δύο αιώνες ενώ τις τελευταίες δεκαετίες έχει γίνει σημαντική σε αρκετούς κλάδους της επιστήμης. Χαρακτηριστικό παράδειγμα αποτελεί ο ρόλος τους στην απόδειξη του *τελευταίου Θεωρήματος του Fermat* την δεκαετία του 80' και 90'.

Σχετικά με το θέμα, υπάρχει τεράστια βιβλιογραφία, τόσο στην θεωρία αριθμών όσο και σε συναφείς τομείς, συμπεριλαμβανομένης της κρυπτογραφίας. Κύριο πλεονέκτημα των ελλειπτικών καμπυλών, αποτελεί η εντυπωσιακή συνεισφορά τους στην ασφάλεια των ψηφιακών υπογραφών και των κρυπτοσυστημάτων δημοσίου κλειδιού.

4.1.1 Ορισμός Ελλειπτικής Καμπύλης

Ορισμός 4.1. Μια *ελλειπτική καμπύλη* πάνω σε ένα σώμα K είναι μια εξίσωση της μορφής:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{4.1}$$

όπου $a_i \in K$, $i = 1, \ldots, 6$ και $\Delta \neq 0$, όπου Δ η *διακρίνουσα* της E που ορίζεται όπως παρακάτω:

$$\begin{aligned}
\Delta &= -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6, \\
b_2 &= a_1^2 + 4a_2, \\
b_4 &= 2a_4 + a_1a_3, \\
b_6 &= a_3^2 + 4a_6, \\
b_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2.
\end{aligned}$$

Η εξίσωση (4.1) ονομάζεται *μακρά μορφή Weierstrass* μιας ελλειπτικής καμπύλης πάνω στο σώμα K και το σύνολο των σημείων της είναι το

$$E(K) = \{(x, y) \in K^2 : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{O_\infty\}$$

όπου $O_\infty = (o_\infty, o_\infty)$ είναι ένα φανταστικό σημείο, το οποίο ονομάζεται *σημείο στο άπειρο*. Η ποσότητα $j = \frac{a_4^3}{\Delta}$ ονομάζεται *αναλλοίωτη* της καμπύλης.

Πόρισμα 4.1. Έστω K ένα σώμα με $char(K) \neq 2, 3$ και E μια ελλειπτική καμπύλη της μορφής (4.1). Θεωρούμε τον μετασχηματισμό σημείων:

$$T : E \to E' \ , \ (x, y) \mapsto (x', y'),$$

με

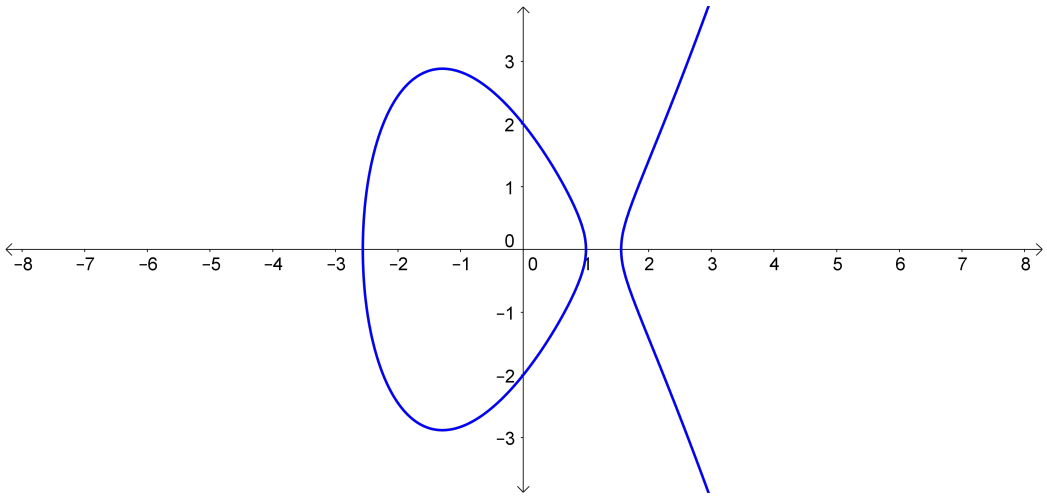
$$x = x' - \tfrac{b_2}{12} \ , \ y = y' - \tfrac{a_1}{2}\left(x' - \tfrac{b_2}{12}\right) - \tfrac{a_3}{2}.$$

Η παραπάνω επιλογή, μετασχηματίζει την ελλειπτική καμπύλη E στην ελλειπτική καμπύλη E', η οποία είναι μια εξίσωση της μορφής:

$$E' : y'^2 = x'^3 + ax' + b \tag{4.2}$$

με $a, b \in K$ και ονομάζεται *βραχεία μορφή Weierstrass* πάνω στο σώμα K. Η διακρίνουσα της καμπύλης είναι η ποσότητα $\Delta = -16\,(4a'^3 + 27b^2)$ με $\Delta \neq 0$ και η αναλλοίωτη αυτής, η ποσότητα $j = \frac{-1729(4a)^3}{\Delta}$.

Παράδειγμα 4.1. Έστω μια ελλειπτική καμπύλη της μορφής (4.2), με εξίσωση $E : y^2 = x^3 - 5x + 4$. Στο επόμενο σχήμα 2.1, φαίνεται η γραφική παράσταση της ελλειπτικής καμπύλης E στο σώμα $\mathbb{R}$.



Σχήμα 2.1 Γραφική παράσταση της E στο $\mathbb{R}$.

Παρατήρηση 4.1. Από την παραπάνω γραφική παράσταση διαπιστώνονται τα εξής:

- Η ελλειπτική καμπύλη είναι συμμετρική ως προς τον άξονα $x'x$. Δηλαδή, αν $(x, y) \in E(K)$, τότε και $(x, -y) \in E(K)$ και το άθροισμα των τεταγμένων τους είναι ίσο με μηδέν.

- Κάθε ευθεία που διέρχεται από δύο τυχαία σημεία της ελλειπτικής καμπύλης, τέμνει την καμπύλη αυτή, σε ένα τρίτο ακόμη σημείο. Το τελευταίο προκύπτει άμεσα από το *Θεώρημα του Bezout* [18].

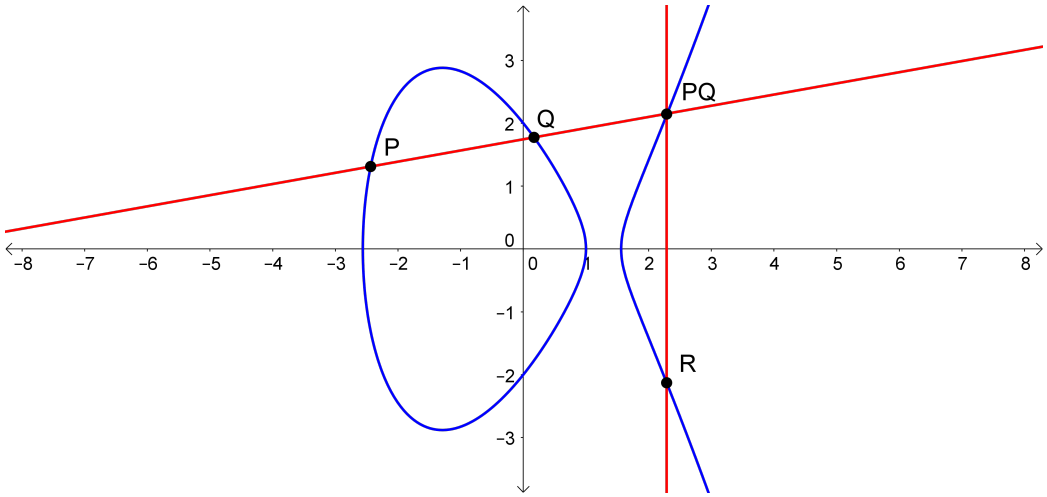
Σύμφωνα με τα παραπάνω, θα συμβολίζουμε με P' το συμμετρικό ενός σημείου P, ενώ το τρίτο σημείο τομής της ελλειπτικής καμπύλη με την ευθεία που διέρχεται από το P και ένα άλλο σημείο Q, θα το συμβολίζουμε με PQ (ή με QP).

4.1.2 Η Ομάδα στις Ελλειπτικές Καμπύλες

Ο κυριότερος λόγος για τον οποίο οι ελλειπτικές καμπύλες παρουσιάζουν ιδιαίτερο ενδιαφέρον, είναι η ιδιότητα των σημείων τους να δομούνται ως ομάδα, με την προσθήκη σε αυτά, μιας κατάλληλα ορισμένης διμελής πράξης. Η πράξη αυτή αρχικά, μπορεί να ερμηνευθεί γεωμετρικά, ως εξής:

Έστω δύο τυχαία σημεία P, Q μίας ελλειπτικής καμπύλης. Η ευθεία που διέρχεται από το σημείο PQ και είναι κάθετη στον άξονα $x'x$, τέμνει την καμπύλη σε δύο ακόμη σημεία. Το ένα από αυτά είναι το O_∞ και το άλλο είναι το συμμετρικό $(PQ)'$ του PQ, το οποίο είναι το αποτέλεσμα της πράξης $P + Q$.

Η προηγούμενη πράξη, παρουσιάζεται στο παρακάτω σχήμα, όπου το σημείο $R = (PQ)'$ είναι το αποτέλεσμα της πράξης $P + Q$.



Σχήμα 2.3 Γεωμετρική απεικόνιση της πράξης $P + Q$.

Ας είναι E μια ελλειπτική καμπύλη της μορφής (4.2) στο σώμα K.

Πρόταση 4.1. Για κάθε $P, Q \in E(K)$ ισχύουν τα παρακάτω:

1. $PP' = O_\infty$.
2. $(P')' = P$.
3. $(PQ)' = P'Q'$.
4. $(P + Q)' = P' + Q'$.

Απόδειξη. Ας είναι $P, Q \in E(K)$.

1. Το σημείο O_∞ βρίσκεται (κατά την αφηρημένη έννοια) στο τέρμα της κάθετης ευθείας που διέρχεται από τα σημεία P, P' και επειδή $O_\infty \in E(K)$, ισχύει $O_\infty = PP'$. Γενικά το σημείο στο άπειρο είναι "συνευθειακό" με όλα τα ζεύγη συμμετρικών σημείων.

2. Προκύπτει άμεσα, από τον ορισμό του συμμετρικού σημείου.

3. Η ευθεία που διέρχεται από τα σημεία P', Q' είναι συμμετρική ως προς τον άξονα $x'x$, με την ευθεία που διέρχεται από τα σημεία P, Q. Επομένως, το σημείο $P'Q'$ θα είναι το συμμετρικό του σημείου PQ.

4. $(P + Q)' = ((PQ)')' = (P'Q')' = P' + Q'$.

□

Θεώρημα 4.1. Έστω $\bullet, +$ δύο διμελείς πράξεις που ορίζονται ως εξής:

$$\bullet : E(K) \times E(K) \to E(K) \ , \ (P, Q) \mapsto PQ$$
$$+ : E(K) \times E(K) \to E(K) \ , \ (P, Q) \mapsto (P \bullet Q)'$$

Το σύνολο $E(K)$ εφοδιασμένο με την πράξη $+$ δομείται σε αβελιανή ομάδα με ουδέτερο στοιχείο το O_∞.

Απόδειξη. Ας είναι $P, Q, R \in E(K)$.

1. Το σύνολο $E(K)$ είναι κλειστό ως προς τις πράξεις $\bullet$ και $+$, δηλαδή, $P \bullet Q \in E(K)$ και $P + Q \in E(K)$. Η κλειστότητα είναι άμεση συνέπεια του ορισμού των δύο αυτών πράξεων.

2. Η διμελής πράξη $\bullet$ είναι αντιμεταθετική. Πράγματι, σύμφωνα με τον ορισμό, το σημείο PQ ταυτίζεται με το QP, αφού και τα δύο αποτελούν το τρίτο σημείο τομής της ευθείας που διέρχεται από τα P, Q με την καμπύλη $E(K)$. Επομένως $PQ = QP \ \Leftrightarrow \ P \bullet Q = Q \bullet P$.

3. Η διμελής πράξη $+$ είναι αντιμεταθετική. Πράγματι $P + Q = (P \bullet Q)' = (Q \bullet P)' = Q + P$.

4. $P + O_\infty = (P \bullet O_\infty)' = (PO_\infty)' = (P')' = P$.

5. $P + P' = (P \bullet P')' = (PP')' = (P'P) = O_\infty$. Επομένως $-P = P'$.

6. $(P + Q) + R = P + (Q + R)$. Η απόδειξη αυτή παραλείπεται, διότι ξεφεύγει από τον σκοπό της διπλωματικής. Ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει στο σύγγραμμα [18].

□

Στο παρακάτω θεώρημα, δίνεται η αναλυτική μορφή του τύπου της διμελής πράξης $+$, ο οποίος ονομάζεται *τύπος της πρόσθεσης στην ομάδα* $(E(K), +)$.

Θεώρημα 4.2. Έστω $P_1 = (x_1, y_1)$ και $P_2 = (x_2, y_2)$ δύο σημεία της $E(K)$ με $P_1, P_2 \neq O_\infty$. Θέτουμε $P_1 + P_2 = P_3 = (x_3, y_3)$. Τότε, έχουμε τα εξής:

1. Αν $x_1 \neq x_2$, τότε
   $x_3 = \lambda^2 - x_1 - x_2 \ , \ y_3 = \lambda(x_1 - x_3) - y_1$ όπου $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.

2. Αν $x_1 = x_2$ αλλά $y_1 \neq y_2$, τότε $P_1 + P_2 = O_\infty$.

3. Αν $P_1 = P_2$ και $y_1 \neq 0$, τότε
   $x_3 = \lambda^2 - 2x_1 \ , \ y_3 = \lambda(x_1 - x_3) - y_1$, όπου $\lambda = \frac{3x_1^2 + a}{2y_1}$.

4. Αν $P_1 = P_2$ και $y_1 = 0$, τότε $P_1 + P_2 = O_\infty$.

4.1.3 Η Ελλειπτική Καμπύλη σε Πεπερασμένο Σώμα

Τα αποτελέσματα της θεωρίας των ελλειπτικών καμπυλών πάνω σε πεπερασμένα σώματα, είναι πλούσια και δεν ελκύουν απλά το ενδιαφέρον από μόνα τους, αλλά αποτελούν τα σημεία εκκίνησης των κρυπτογραφικών εφαρμογών που αναπτύσσονται στη συνέχεια.

Ένα πεπερασμένο σώμα με q στοιχεία, συμβολίζεται με $\mathbb{F}_q$. Καλούμε τάξη της ελλειπτικής καμπύλης πάνω στο σώμα αυτό, το πλήθος των στοιχείων της ομάδας $E(\mathbb{F}_q)$. Όταν $q = p^n$ για κάποιο $p \in \mathbb{P}$ και φυσικό $n \geq 1$, τότε έχουμε το ακόλουθο θεώρημα:

Θεώρημα 4.3. Για κάθε πρώτο αριθμό p και κάθε θετικό ακέραιο $n \geq 1$ υπάρχει ένα μοναδικό πεπερασμένο σώμα $\mathbb{F}$, κατά προσέγγιση ισομορφισμού, με p^n στοιχεία. Αντίστροφα, κάθε πεπερασμένο σώμα είναι αυτής της μορφής.

Το θεώρημα αυτό, οφείλεται στον *E.Galois*.

Στο επόμενο θεώρημα, δίνεται ο τύπος υπολογισμού της τάξης μιας ελλειπτικής καμπύλης πάνω σε ένα πεπερασμένο σώμα. Ο τύπος αυτός είναι εύχρηστος, για πεπερασμένα σώματα μικρής σχετικά τάξης.

Θεώρημα 4.4. Έστω E μια ελλειπτική καμπύλη επί του $\mathbb{F}_q$, η οποία έχει τύπο $E : y^2 = x^3 + ax + b = f(x)$. Τότε,

$$|E(\mathbb{F}_q)| = q + 1 + \sum_{x \in \mathbb{F}_q} \left(\frac{f(x)}{\mathbb{F}_q} \right), \tag{4.3}$$

όπου για τυχαίο $x \in \mathbb{F}_q$, ισχύει:

$$\left(\frac{f(x)}{\mathbb{F}_q} \right) = \begin{cases} +1 & , \text{ αν υπάρχει } y \in \mathbb{F}_q - \{0\} \text{ με } y^2 = f(x), \\ -1 & , \text{ αν } \forall\ y \in \mathbb{F}_q \text{ έχουμε } y^2 \neq f(x), \\ 0 & , \text{ αν } f(x) = 0. \end{cases}$$

Ο συμβολισμός της μορφής $\left(\frac{a}{b}\right)$, προέρχεται από τα σύμβολα *Legendre*.

Σύμφωνα με το προηγούμενο θεώρημα και την σχέση (2.1) προκύπτει το επόμενο πόρισμα:

Πόρισμα 4.2. Έστω E μια ελλειπτική καμπύλη επί του $\mathbb{Z}_p$ με $p \equiv 3 \bmod 4$ και τύπο $E : y^2 \equiv f(x) \bmod p$, με $f(x) = x^3 + ax + b$. Τότε,

$$|E(\mathbb{Z}_p)| = p + 1 + \sum_{x \in \mathbb{Z}_p} \left(\frac{f(x)}{\mathbb{Z}_p} \right) \tag{4.4}$$

όπου για τυχαίο $x \in \mathbb{Z}_p$, ισχύει:

$$\left(\frac{f(x)}{\mathbb{Z}_p}\right) = \begin{cases} +1 & , \text{ αν } f(x) \equiv f(x)^{(p+1)/2} \bmod p \text{ και } f(x) \not\equiv 0 \bmod p, \\ -1 & , \text{ αν } f(x) \not\equiv f(x)^{(p+1)/2} \bmod p, \\ 0 & , \ f(x) \equiv 0 \bmod p. \end{cases}$$

Παράδειγμα 4.2. Έστω η ελλειπτική καμπύλη $E : y^2 \equiv f(x) \bmod 7$ με $f(x) = x^3 + 2x + 4$. Τα σημεία του συνόλου $E(\mathbb{Z}_7)$, προκύπτουν από τον επόμενο πίνακα:

x	$f(x) \bmod 7$	$f(x)^{(p+1)/2} \bmod 7$	$\left(\frac{f(x)}{\mathbb{Z}_7}\right)$	$y = \pm f(x)^{(p+1)/4} \bmod 7$
0	4	4	$+1$	± 2
1	0	0	0	0
2	2	2	$+1$	± 4
3	2	2	$+1$	± 4
4	6	1	-1	$-$
5	6	1	-1	$-$
6	1	1	$+1$	± 6

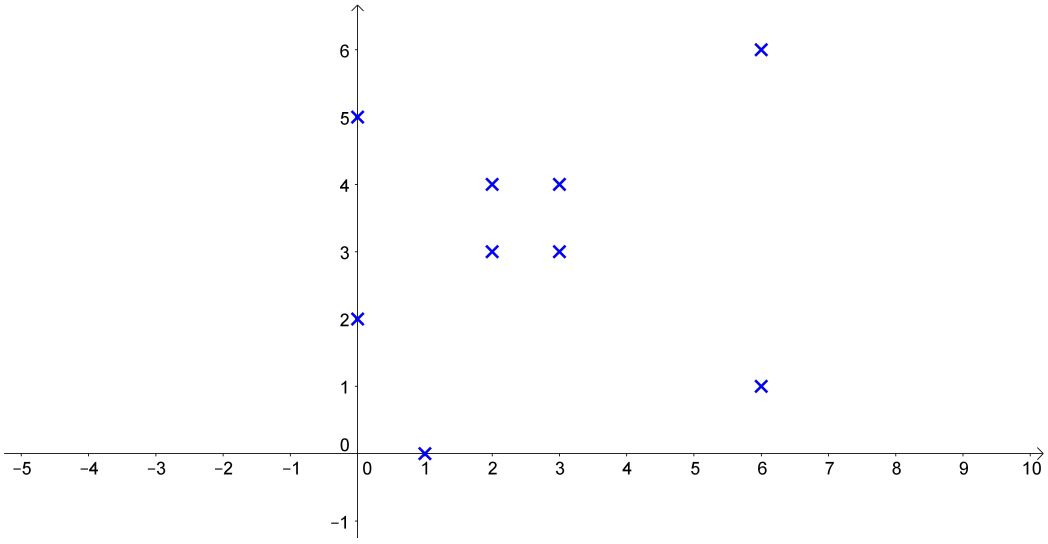
Επομένως

$$E(\mathbb{Z}_7) = \{(0,2),(0,5),(1,0),(2,3),(2,4),(3,3),(3,4),(6,1),(6,6)\} \cup \{O_\infty\},$$

και

$$\begin{aligned} |E(\mathbb{Z}_7)| &= 7 + 1 + \left(\tfrac{f(0)}{\mathbb{Z}_7}\right) + \cdots + \left(\tfrac{f(6)}{\mathbb{Z}_7}\right) \\ &= 7 + 1 + 1 + 0 + 1 + 1 - 1 - 1 + 1 \\ &= 10. \end{aligned}$$

Στο παρακάτω Σχήμα 2.2, φαίνεται η γραφική παράσταση της ελλειπτικής καμπύλης E επί του $\mathbb{Z}_7$.



Σχήμα 2.2 Γραφική παράσταση της E επί του $\mathbb{Z}_7$.

Στο επόμενο θεώρημα, γνωστό ως *Θεώρημα του Hasse*, αναφέρεται ότι η τιμή της τάξης μιας ελλειπτικής καμπύλης πάνω σε ένα πεπερασμένο σώμα, περιορίζεται από μια ελάχιστη και μια μέγιστη τιμή, το μέγεθος των οποίων, εξαρτάται από την τιμή της τάξης του πεπερασμένου αυτού σώματος.

Θεώρημα 4.5. Έστω E μια ελλειπτική καμπύλη επί του F_q. Τότε ισχύει:

$$q + 1 - 2\sqrt{q} \leq |E(\mathbb{F}_q)| \leq q + 1 + 2\sqrt{q}. \tag{4.5}$$

Έστω J_q το σύνολο όλων των ελλειπτικών καμπυλών επί του $\mathbb{F}_q$, με τάξη στο διάστημα $\left[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}\right]$. Στο επόμενο θεώρημα, αναφέρονται οι προϋποθέσεις ύπαρξης μιας ελλειπτικής καμπύλης $E \in J_q$ τάξης k, όταν $q = p^n$, για κάποιο $p \in \mathbb{P}$ και φυσικό $n \geq 1$.

Θεώρημα 4.6. Έστω $q = p^n$, για κάποιο $p \in \mathbb{P}$ και φυσικό $n \geq 1$. Υπάρχει ελλειπτική καμπύλη $E \in J_q$, τέτοια ώστε $|E(\mathbb{F}_q)| = q + 1 - a$, αν και μόνο αν, $|a| \leq 2\sqrt{q}$ και ο a ικανοποιεί ένα από τα παρακάτω κριτήρια:

1. $\mu\kappa\delta\,(a, p) = 1$.
2. n είναι άρτιος και $a = \pm 2\sqrt{q}$.
3. n είναι άρτιος, $p \not\equiv 1 \bmod 3$ και $a = \pm 2\sqrt{q}$.
4. n είναι περιττός, $p \in \{2, 3\}$ και $a = \pm p^{\frac{n+1}{2}}$
5. n είναι άρτιος, $p \not\equiv 1 \bmod 4$ και $a = 0$.
6. n είναι περιττός και $a = 0$.

Ο αριθμός a ονομάζεται *ίχνος του Frobenius*.

Το 1985 ο Schoof πρώτος [38], παρουσίασε εναν αλγόριθμο πολυωνυμικής πολυπλοκότητας χρόνου, ο οποίος υπολογίζει το πλήθος των σημείων μιας ελλειπτικής καμπύλης πάνω σε ένα πεπερασμένο σώμα $\mathbb{F}_p$, όπου p πολύ μεγάλος πρώτος αριθμός.

Ορισμός 4.2. Μία ελλειπτική καμπύλη E επί του $\mathbb{F}_p$, ονομάζεται *υπεριδιάζουσα*, αν και μόνο αν, $|E(\mathbb{F}_p)| = p + 1$, όπου $p \in \mathbb{P}$ με $p \geq 5$.

Ορισμός 4.3. Μία ελλειπτική καμπύλη E επί του $\mathbb{F}_p$, ονομάζεται *μη ομαλή*, αν και μόνο αν, $|E(\mathbb{F}_p)| = p$, όπου $p \in \mathbb{P}$.

Ορισμός 4.4. Έστω $m \in \mathbb{Z}$ και $P \in E(\mathbb{F}_q)$. Ονομάζουμε *πολλαπλασιασμό του ακέραιου m με το P* και το συμβολίζουμε με mP, την πρόσθεση του P με τον εαυτό του m φορές. Δηλαδή,

$$mP = \begin{cases} \underbrace{P + \cdots + P}_{m \text{ φορές}} & , \text{ αν } m > 0 \\ \underbrace{(-P) + \cdots + (-P)}_{m \text{ φορές}} & , \text{ αν } m < 0 \\ O_\infty & , \text{ αν } m \in \{0, n, 2n, \ldots\}, n \in \mathbb{N} \end{cases}$$

όπου n η τάξη του σημείου P.

Για τον πολλαπλασιασμό ενός ακέραιου m με ένα σημείο $P \in E(\mathbb{F}_q)$, έχουν αναπτυχθεί διάφορες μέθοδοι. Η απλούστερη (και παλαιότερη) μέθοδος, στηρίζεται στον διαδοχικό διπλασιασμό του σημείου P και περιγράφεται με τον παρακάτω αλγόριθμο:

Αλγόριθμος 4.1. mP: Διαδοχικός Διπλασιασμός.

Είσοδος: Ακέραιος m και σημείο $P \in E(\mathbb{F}_q)$.
Έξοδος: Σημείο $Q = mP$.

1. $Q \leftarrow O_\infty$.
2. Όσο $m > 0$ επανέλαβε:
   (α′) Αν $m \equiv 1 \mod 2$ τότε:
      i. $Q \leftarrow Q + P$,
      ii. $m \leftarrow m - 1$.
   (β′) $P \leftarrow P + P$.
   (γ′) $m \leftarrow m/2$.
3. Επίστρεψε Q.

Αν ο ακέραιος m δίνεται με την δυαδική του μορφή και $l_2(m) = k$, τότε ο προηγούμενος αλγόριθμος, μετατρέπεται στον επόμενο αλγόριθμο:

Αλγόριθμος 4.2. mP : Δυαδική Μέθοδος.

Είσοδος: Ακέραιος $m = \sum_{i=1}^{k} m_i 2^{k-i}$, $m_i \in \{0, 1\}$ και σημείο $P \in E(\mathbb{F}_q)$.
Έξοδος: Σημείο $Q = mP$.

1. $Q \leftarrow O_\infty$.
2. Για $i = 1$ μέχρι k
   (α′) $Q \leftarrow Q + Q$,
   (β′) Αν $k_i = 1$ τότε $Q \leftarrow Q + P$.
3. Επίστρεψε Q.

Η πολυπλοκότητα χρόνου αυτών των αλγορίθμων είναι: $O(\log_2 m)$ πράξεις μέσα στην ομάδα της ελλειπτικής καμπύλης.

Με βάση τα Θεωρήματα 2.9 και 4.5, μπορούμε να υπολογίσουμε την τάξη ενός σημείου μιας ελλειπτικής καμπύλης E επί του $\mathbb{F}_q$ καθώς και την ίδια την τάξη της ελλειπτικής καμπύλης.

Σύμφωνα με αυτά τα θεωρήματα, αν $P \in E(\mathbb{F}_q)$, η τάξη του σημείου αυτού είναι ο μικρότερος θετικός ακέραιος k για τον οποίο $kP = O_\infty$. Ο ακέραιος αυτός, διαιρεί την τάξη της ελλειπτικής καμπύλης E. Επομένως, αν γνωρίζουμε την πρωτογενή ανάλυση της τάξης αυτής (και συνεπώς τους διαιρέτες της), ελέγχουμε ποιος από αυτούς, ικανοποιεί την σχέση $xP = O_\infty$, όπου x διαιρέτης. Η τάξη του σημείου P, θα είναι ο μικρότερος από τους διαιρέτες που επαληθεύουν την σχέση αυτή.

Αντίστροφα, αν είναι γνωστές οι τάξεις ενός ή περισσότερων σημείων της ελλειπτικής καμπύλης E, μπορούμε σχετικά εύκολα, να υπολογίσουμε και την τάξη της ελλειπτικής καμπύλης E. Διακρίνουμε τις παρακάτω περιπτώσεις:

1. Αν k η τάξη ενός σημείου $P \in E(\mathbb{F}_q)$, αναζητούμε εναν ακέραιο l, τέτοιον ώστε:
   $$k \cdot l \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]\,,$$
   με
   $$kl_1 < q + 1 - 2\sqrt{q},\ \forall\ l_1 < l \text{ και } kl_2 > q + 1 + 2\sqrt{q},\ \forall\ l_2 > l.$$
   Αν υπάρχει αυτός ο ακέραιος, τότε $|E(\mathbb{F}_q)| = k \cdot l$.
2. Αν k_i είναι οι τάξεις n σημείων $P_i \in E(\mathbb{F}_q)$, $i = 1, \ldots, n$, τότε αναζητούμε έναν ακέραιο l και εναν ελάχιστο ακέραιο m με $1 < m \leq n$, έτσι ώστε:
   $$k_1 \ldots k_m \cdot l \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]\,,$$

με

$k_1 \dots k_m \cdot l_1 < q+1-2\sqrt{q},\ \forall\, l_1 < l$ και $k_1 \dots k_m \cdot l_2 > q+1+2\sqrt{q},\ \forall\, l_2 > l.$

Αν υπάρχει αυτός ο ακέραιος, τότε $|E(\mathbb{F}_q)| = k_1 \dots k_m \cdot l.$

Παράδειγμα 4.3. Έστω η ελλειπτική καμπύλη $E : y^2 = x^3 + 7x + 12$ επί του $\mathbb{F}_{103}$ και $(-1, 2)$, $(19, 0)$ δύο σημεία της.

Έστω $|E(\mathbb{F}_{103})| = 104$. Έχουμε, $104 = 13 \cdot 2^3$ και επομένως όλοι οι διαιρέτες του 104 προκύπτουν από τους ακέραιους $2, 4, 8, 13, 26$. Παρατηρούμε οτι οι αριθμοί 13 και 2 είναι οι μικρότεροι ακέραιοι με $13(-1, 2) = O_\infty$ και $2(19, 0) = O_\infty$ αντίστοιχα.

Αντίστροφα, έστω ότι οι τάξεις των δύο σημείων είναι 13 και 2 αντίστοιχα. Τότε,

$$84 < 104 = 13 \cdot 2 \cdot 4 < 124,$$

ενώ $13 \cdot 2 \cdot 5 = 130 > 124$ και $13 \cdot 2 \cdot 3 = 78 < 84$. Επομένως $|E(\mathbb{F}_{103})| = 104$.

Κεφάλαιο 5

Συναρτήσεις Συμπύκνωσης

Οι κρυπτογραφικές συναρτήσεις κατακερματισμού, αποτελούν τη βάση της ασφάλειας του σημερινού ψηφιακού περιβάλλοντος και σχεδιάστηκαν για να παρέχουν ισχυρή εμπιστοσύνη και ασφάλεια σε πολλά κρυπτογραφικά συστήματα. Στο κεφάλαιο αυτό, θα εισάγουμε την έννοια, καθώς και τις ιδιότητες των συναρτήσεων κατακερματισμού, με τις οποίες ενισχύεται η ασφάλεια των σχημάτων ψηφιακών υπογραφών.

5.1 Συνάρτηση Κατακερματισμού

Οι ακολουθίες (τυπογραφικών) χαρακτήρων αποτελούν θεμελιώδη δομικά στοιχεία της επιστήμης υπολογιστών. Το αλφάβητο επί του οποίου ορίζονται οι ακολουθίες αυτές, μπορεί να ποικίλλει ανάλογα με την εφαρμογή. Για τους σκοπούς αυτής της διπλωματικής, ισχύει ο ακόλουθος ορισμός:

Ορισμός 5.1. Ένα πεπερασμένο σύνολο από σύμβολα ονομάζεται *αλφάβητο* και συμβολίζεται με

$$\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \ , \ n \in \mathbb{N} - \{0\} \, .$$

Τα σύμβολα σ_i, $i = 1 \ldots, n$ μπορεί να είναι κάθε είδους, όπως: γράμματα, αριθμοί, ειδικά σύμβολα (πχ #,&,!) κ.α.

Ας είναι $\Sigma_A = \{a_1, \ldots, a_k\}$ και $\Sigma_B = \{b_1, \ldots, b_l\}$ δύο αλφάβητα (όχι απαραίτητα διαφορετικά) τα οποία αποτελούνται από k και l διακεκριμένα σύμβολα, αντίστοιχα. Για κάθε $r, t > 1$, ορίζουμε τα σύνολα:

$$\Sigma_A^r = \{x_1 \ldots x_r : x_i \in \Sigma_A \ , \ i = 1, \ldots, r\}$$

$$\Sigma_B^t = \{x_1 \dots x_t : x_i \in \Sigma_B \ , \ i = 1, \dots, t\}$$

και

$$\Sigma_A^* = \bigcup_{i=1}^{\infty} \Sigma_A^i.$$

Ορισμός 5.2. Μια *συνάρτηση κατακερματισμού* είναι μια συνάρτηση της μορφής:

$$\mathcal{H} : \Sigma_A^* \longrightarrow \Sigma_B^n \ , \ n \in \mathbb{N} - \{0\} .$$

Η συνάρτηση αυτή, δέχεται ως είσοδο μια συμβολοσειρά από το αλφάβητο Σ_A αυθαιρέτου μήκους και επιστρέφει μια συμβολοσειρά από το αλφάβητο Σ_A μήκους n.

Έστω Σ το αλφάβητο το οποίο αποτελείται από τα πεζά γράμματα του λατινικού αλφάβητου και τους αριθμούς $0, 1, \dots, 9$. Έστω ένα σύμβολο $\sigma \in \Sigma$. Θεωρούμε την συνάρτηση

$$N_\sigma : \Sigma^* \longrightarrow \mathbb{N} \ , \ x \longmapsto N_\sigma (x)$$

η οποία επιστρέφει το πλήθος $N_\sigma (x)$ των εμφανίσεων του συμβόλου σ μέσα στην συμβολοσειρά x. πχ: έστω το κείμενο x:

national security agency use hash functions on digital signatures.

Τότε, $N_o (x) = 3$ και $N_i (x) = 6$. Οι συναρτήσεις N_σ , $\sigma \in \Sigma$ δεν αποτελούν συναρτήσεις κατακερματισμού, ωστόσο μπορούν να χρησιμοποιηθούν για την κατασκευή αυτών, όπως στα επόμενα παραδείγματα:

Παράδειγμα 5.1. Έστω ένα σύμβολο $\sigma \in \Sigma$. Θεωρούμε την συνάρτηση

$$\mathcal{H}_\sigma : \Sigma^* \longrightarrow \{0, 1\} \ , \ x \longmapsto N_\sigma (x) \bmod 2$$

Η συνάρτηση $\mathcal{H}_\sigma$ είναι μια συνάρτηση κατακερματισμού. πχ: για το κείμενο x του προηγούμενου παραδείγματος, ισχύει: $\mathcal{H}_o (x) = 1$, $\mathcal{H}_i (x) = 0$.

Παράδειγμα 5.2. Θεωρούμε την συνάρτηση

$$\mathcal{H}_A : \Sigma^* \longrightarrow \{0, 1\}^{26} \ , \ x \longmapsto \mathcal{H}_a (x) \left\| \mathcal{H}_b (x) \right\| \dots \left\| \mathcal{H}_z (x) \right. .$$

Η συνάρτηση $\mathcal{H}_A$ είναι μια συνάρτηση κατακερματισμού, η οποία επιστρέφει συμβολοσειρές μήκους 26 αποτελούμενες από τα σύμβολα 0 και 1. πχ:

$$\begin{aligned} \mathcal{H}_A (\text{artillery fire}) &= 10000100000000000101000010 \\ \mathcal{H}_A (\text{coordinates}) &= 10111000100001000111000000 \end{aligned}$$

Το σύμβολο $\left\|\right.$, ενώνει συμβολοσειρές, οι οποίες βρίσκονται εκατέρωθέν του.

Παράδειγμα 5.3. Θεωρούμε την συνάρτηση

$$\mathcal{H}_B : \Sigma^* \longrightarrow \{0,1\}^{10} \ , \ x \longmapsto \mathcal{H}_0(x) \left\| \mathcal{H}_1(x) \right\| \ldots \left\| \mathcal{H}_9(x) \right.$$

Η συνάρτηση $\mathcal{H}_B$ είναι μια συνάρτηση κατακερματισμού, η οποία επιστρέφει συμβολοσειρές μήκους 10 αποτελούμενες από τα σύμβολα 0 και 1. πχ:

$$\mathcal{H}_B(8097443\ 4123649\ 472) \ = \ 1100101010.$$

Παράδειγμα 5.4. Θεωρούμε την συνάρτηση

$$\mathcal{H} : \Sigma^* \longrightarrow \{0,1\}^{36} \ , \ x \longmapsto \mathcal{H}_A(x) \left\| \mathcal{H}_B(x) \right.$$

Η συνάρτηση $\mathcal{H}$ είναι μια συνάρτηση κατακερματισμού, η οποία επιστρέφει συμβολοσειρές μήκους 36 αποτελούμενες από τα σύμβολα 0 και 1. πχ: έστω x:

$$\text{artillery fire coordinates } 8097443\ 4123649\ 472$$

τότε

$$\mathcal{H}(x) = 001111001000010000100000101100101010.$$

Παράδειγμα 5.5. Έστω το αλφάβητο $\Sigma = \{a, b\}$. Η συνάρτηση

$$\mathcal{H} : \Sigma^* \longrightarrow \Sigma \ , \ \ x \longmapsto \begin{cases} a & , \quad N_a(x) > N_b(x) \\ b & , \quad N_a(x) \leq N_b(x) \end{cases}$$

είναι μια συνάρτηση κατακερματισμού, η οποία μετασχηματίζει την συμβολοσειρά x στο σύμβολο a, αν στην συμβολοσειρά x το πλήθος των συμβόλων a είναι μεγαλύτερο από το πλήθος των συμβόλων b ή στο σύμβολο b, σε αντίθετη περίπτωση.

Γενικά, οι συναρτήσεις κατακερματισμού, μπορούν να δημιουργηθούν και με συναρτήσεις συμπύκνωσης.

Ορισμός 5.3. Μία *συνάρτηση συμπύκνωσης* είναι μια συνάρτηση της μορφής

$$\mathcal{H} : \Sigma_A^l \longrightarrow \Sigma_B^s \ , \ l, s \in \mathbb{N} \ , \ l > s$$

η οποία δέχεται ως είσοδο μια συμβολοσειρά μήκους $l > s$ και την μετατρέπει σε μια συμβολοσειρά μήκους s.

Παράδειγμα 5.6. Έστω $\Sigma = \{0,1\}$ το δυαδικό αλφάβητο και ένας ακέραιος $l > 1$. Η συνάρτηση

$$\mathcal{H} : \Sigma^l \longrightarrow \Sigma \ , \ x_1 x_2 \cdots x_l \longmapsto x_1$$

είναι μία συνάρτηση συμπύκνωσης, η οποία μετασχηματίζει την λέξη $x_1 x_2 \cdots x_l$ στο πρώτο της γράμμα x_1.

Παρατήρηση 5.1. Για τα Παραδείγματα 5.5 και 5.6 αναφέρουμε τα εξής:

- Ο μετασχηματισμός της λέξης $x_1 x_2 \cdots x_l$ σε x_1 είναι μια απλή και γρήγορη διαδικασία. Επιπλέον, αν γνωρίζουμε την τιμή x_1 της συνάρτησης $\mathcal{H}$ για κάποιο $x \in \{0,1\}^l$, θα υπάρχουν πεπερασμένου πλήθους λέξεις μήκους l που το πρώτο τους γράμμα να είναι το x_1 και έτσι η εύρεση ενός x για το οποίο $\mathcal{H}(x) = x_1$, είναι αρκετά εύκολη.

- Με την γνώση της τιμής a της συνάρτησης $\mathcal{H}$ για κάποιο $x \in \{a,b\}^*$, υπάρχουν άπειρες λέξεις πεπερασμένου μήκους όπου το πλήθος των a σε κάθε μία από αυτές να είναι μεγαλύτερο από το πλήθος των b. Έτσι και εδώ, η εύρεση ενός x για το οποίο $\mathcal{H}(x) = a$, είναι αρκετά εύκολη.

- Ο μετασχηματισμός της λέξης x σε a, απαιτεί περισσότερο χρόνο, καθώς θα πρέπει αρχικά, να μετρηθεί η εμφάνιση των a και b αντίστοιχα, μέσα στη λέξη x.

Ορισμός 5.4. Μια συνάρτηση $f : X \longrightarrow Y$, καλείται *μοναδικής κατεύθυνσης*, αν υπάρχει αλγόριθμος πολυωνυμικού χρόνου, που να υπολογίζει τις τιμές $f(x)$ για κάθε $x \in X$ και αν είναι "πρακτικά" αδύνατον, σχεδόν για κάθε $y \in f(X)$, να βρεθεί ένα $x \in X$ τέτοιο ώστε $y = f(x)$.

Οι συναρτήσεις συμπύκνωσης οι οποίες είναι μοναδικής κατεύθυνσης, θεωρούνται κατάλληλες για να χρησιμοποιηθούν σε κρυπτογραφικά συστήματα.

Παράδειγμα 5.7. Έστω p ένας αρκετά μεγάλος πρώτος αριθμός και g μια αρχική ρίζα mod p. Θεωρούμε την συνάρτηση

$$\mathcal{H} : \{0,1,\ldots,p-2\} \to Z_p^* \quad , \quad x \mapsto g^{-x}.$$

Οι τιμές της $\mathcal{H}$ υπολογίζονται σε πολυωνυμικό χρόνο, ενώ δεν είναι ακόμη γνωστό, αν υπάρχει αλγόριθμος που να υπολογίζει την $\mathcal{H}^{-1}$. Η παραπάνω συνάρτηση επομένως, μπορεί να χρησιμοποιηθεί ως συνάρτηση μοναδικής κατεύθυνσης.

Ορισμός 5.5. Έστω $\mathcal{H}$ μια συνάρτηση συμπύκνωσης η οποία είναι και μοναδικής κατεύθυνσης.Ένα ζεύγος (x_1, x_2) ονομάζεται *σύμπτωση* της $\mathcal{H}$, αν $x_1 \neq x_2$ και $\mathcal{H}(x_1) = \mathcal{H}(x_2)$, ενώ αν είναι αδύνατον υπολογιστικά να βρεθεί μια σύμπτωση, η $\mathcal{H}$ ονομάζεται *ελεύθερης σύμπτωσης*.

Μερικές από τις σημαντικότερες συναρτήσεις ελεύθερης σύμπτωσης, περιγράφονται στην επόμενη ενότητα.

5.2 Συναρτήσεις Κατακερματισμού SHA

Οι συναρτήσεις κατακερματισμού SHA (Secure Hash Algorithm) άρχισαν να δημοσιεύονται από το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) των ΗΠΑ το 1993. Μέχρι σήμερα, η οικογένεια αυτή απαριθμείται από τις κατηγορίες συναρτήσεων SHA-0, SHA-1, SHA-2 και SHA-3.

Οι κατηγορίες αυτές προέκυψαν σταδιακά, για να καλύψουν ειδικές απαιτήσεις ασφαλείας σε κρυπτογραφικά σχήματα ή για να βελτιώσουν τυχόν αδυναμίες παλαιότερων προδιαγραφών. Όλες οι συναρτήσεις, χρησιμοποιούνται αλγοριθμικά, για την ταυτοποίηση μηνυμάτων καθώς και για την παραγωγή τυχαίων αριθμών ή bits. Με τις συναρτήσεις κατακερματισμού SHA, η υπαρκτή πλαστογράφηση μιας ψηφιακής υπογραφής, καθίσταται πρακτικά αδύνατη.

Στην ενότητα αυτή, θα αναφερθούμε σε δύο δημοφιλείς κατηγορίες, την SHA-1 και SHA-2, περιγράφοντας την αρχιτεκτονική λειτουργίας τους. Όλες οι συναρτήσεις που ανήκουν σε αυτές τις κατηγορίες, σχεδιάστηκαν από την NSA και χρησιμοποιούνται από τα σχήματα ψηφιακών υπογραφών DSA και ECDSA.

5.2.1 SHA-1

Αρχικά, θα περιγράψουμε την λειτουργία όλων των παραμέτρων που χρησιμοποιούνται από την συνάρτηση SHA-1.

Ορισμός Συμβόλων

$\wedge : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ λογική πράξη AND με τύπο:

$$\begin{array}{rcl} (0,0) & \mapsto & 0 \quad , \quad k=1 \\ (1,0) & \mapsto & 0 \quad , \quad k=1 \\ (0,1) & \mapsto & 0 \quad , \quad k=1 \\ (1,1) & \mapsto & 1 \quad , \quad k=1 \\ (x_1 \ldots x_k, y_1 \ldots y_k) & \mapsto & x_1 \wedge y_1 \ldots x_k \wedge y_k \quad , \quad k=32 \end{array} \tag{5.1}$$

$\vee : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ λογική πράξη OR με τύπο:

$$\begin{array}{rcl} (0,0) & \mapsto & 0 \quad , \quad k=1 \\ (1,0) & \mapsto & 1 \quad , \quad k=1 \\ (0,1) & \mapsto & 1 \quad , \quad k=1 \\ (1,1) & \mapsto & 1 \quad , \quad k=1 \\ (x_1 \dots x_k, y_1 \dots y_k) & \mapsto & x_1 \vee y_1 \dots x_k \vee y_k \quad , \quad k=32 \end{array} \tag{5.2}$$

$\oplus : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$ λογική πράξη XOR με τύπο:

$$\begin{array}{rcl} (0,0) & \mapsto & 0 \quad , \quad k=1 \\ (1,0) & \mapsto & 1 \quad , \quad k=1 \\ (0,1) & \mapsto & 1 \quad , \quad k=1 \\ (1,1) & \mapsto & 0 \quad , \quad k=1 \\ (x_1 \dots x_k, y_1 \dots y_k) & \mapsto & x_1 \oplus y_1 \dots x_k \oplus y_k \quad , \quad k=32 \end{array} \tag{5.3}$$

$\neg : \{0,1\}^k \to \{0,1\}^k$ λογική πράξη NOT με τύπο:

$$\begin{array}{rcl} 0 & \mapsto & 1 \quad , \quad k=1 \\ 1 & \mapsto & 0 \quad , \quad k=1 \\ x_1 \dots x_k & \mapsto & \neg x_1 \dots \neg x_k \quad , \quad k=32 \end{array} \tag{5.4}$$

$+ : \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{32}$ πρόσθεση mod 2^{32} με τύπο:

$$(x,y) \mapsto z \tag{5.5}$$

όπου z το αποτέλεσμα του αλγορίθμου 3.2 με είσοδο x, y.

$\ll$: $\{0,1\}^{32} \to \{0,1\}^{32}$ αριστερή λογική μετατόπιση με τύπο:

$$x_1 \dots x_{32} \mapsto (x_1 \dots x_{32} \ll) = x_2 \dots x_{32} \| 0 \tag{5.6}$$

όπου $(x_1 \dots x_{32} \ll n) = x_{1+n} \dots x_{32} \| \underbrace{0 \dots 0}_{n \text{ φορές}}$.

$\gg$: $\{0,1\}^{32} \to \{0,1\}^{32}$ δεξιά λογική μετατόπιση με τύπο:

$$x_1 \dots x_{32} \mapsto (x_1 \dots x_{32} \gg) = 0 \| x_1 \dots x_{31} \tag{5.7}$$

όπου $(x_1 \dots x_{32} \gg n) = \underbrace{0 \dots 0}_{n \text{ φορές}} \| x_1 \dots x_{32-n}$.

$ROTL^n : \{0,1\}^{32} \to \{0,1\}^{32}$ αριστερή περιστροφή με τύπο:

$$x \mapsto ROTL^n(x) = (x \ll n) \vee (x \gg 32 - n) \tag{5.8}$$

όπου $0 \leq n < 32$.

Στον επόμενο πίνακα, δίνονται ορισμένα παραδείγματα πράξεων με τα παραπάνω σύμβολα.

Παράδειγμα 5.8.

Σύμβολο	Πράξη	Αποτέλεσμα
$\wedge$	5c2f3a17 $\wedge$ 27d1b8e9	04013801
$\vee$	5c2f3a17 $\vee$ 27d1b8e9	7fffbaff
$\oplus$	5c2f3a17 $\oplus$ 27d1b8e9	7bfe82fe
$\neg$	$\neg$5c2f3a17	a3d0c5e8
$+$	5c2f3a17 $+$ 27d1b8e9	8400f3f0
$\ll$	5c2f3a17 $\ll$	b85e742e
$\gg$	5c2f3a17 $\gg$	2e179d0b
$\ll 2$	5c2f3a17 $\ll 2$	70bce85c
$\gg 2$	5c2f3a17 $\gg 2$	170bce85
$\gg 30$	5c2f3a17 $\gg 30$	00000001
$\ll 30$	5c2f3a17 $\ll 30$	c0000000
$ROTL^2$	(5c2f3a17 $\ll 2$) $\vee$ (5c2f3a17 $\gg 30$)	70bce85d

Ορισμός Λογικών Συναρτήσεων

$f_t : \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{32}$ με τύπο:

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) & , \quad t = 0, \ldots, 19 \\ Parity(x,y,z) = x \oplus y \oplus z & , \quad t = 20, \ldots, 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & , \quad t = 40, \ldots, 59 \\ Parity(x,y,z) = x \oplus y \oplus z & , \quad t = 60, \ldots, 79 \end{cases}$$

Ορισμός Σταθερών Συναρτήσεων

$$K_t = \begin{cases} \text{5a827999} & , \quad t = 0, \ldots, 19 \\ \text{6ed9eba1} & , \quad t = 20, \ldots, 39 \\ \text{8f1bbcdc} & , \quad t = 40, \ldots, 59 \\ \text{ca62c1d6} & , \quad t = 60, \ldots, 79 \end{cases}$$

$$\begin{aligned} H_0^{(0)} &= 67452301 \\ H_1^{(0)} &= efcdab89 \\ H_2^{(0)} &= 98badcfe \\ H_3^{(0)} &= 10325476 \\ H_4^{(0)} &= c3d2e1f0 \end{aligned}$$

Είσοδος και Διαμόρφωση Μηνύματος

Επέκταση Μηνύματος Η συνάρτηση κατακερματισμού SHA-1 δέχεται ως είσοδο ένα μήνυμα M στην $2-$αδική του μορφή, με μήκος n bits όπου $0 \leq n < 2^{64}$. Το μήκος n πρέπει να είναι ακέραιο πολλαπλάσιο του 512. Αν αυτό δεν ισχύει, τότε το μήνυμα διαμορφώνεται ανάλογα, ως εξής: Πραγματοποιείται η δεξιά προσκόλληση του δυαδικού ψηφίου 1 στο μήνυμα και ακολούθως προσκαλούνται ακόμη k δυαδικά ψηφία 0 όπου k η μικρότερη θετική λύση της ισοτιμίας $k + n \equiv 447 \bmod 512$. Στη συνέχεια, πραγματοποιείται η δεξιά προσκόλληση $64 - l_2(n)$ δυαδικών ψηφίων 0 ακολουθούμενα από την $2-$αδική μορφή του ακεραίου n. Το νέο μήνυμα $M_{\text{νέο}}$ αποτελείται τώρα από $n + 1 + k + 64$ bits.

Παράδειγμα 5.9. Έστω το μήνυμα $M = 01100110000110110010101110$. Για το μήνυμα αυτό έχουμε $n = 26$ και $k = 421$ αφού $421 + 26 \equiv 447 \bmod 512$. Επομένως,

$$M_{\text{νέο}} = \underbrace{01100110000110110010101110}_{26 \text{ bits}} \| 1 \| \underbrace{0 \ldots 0}_{421 \text{ φορές}} \| \overbrace{\underbrace{0 \ldots 0}_{59 \text{ φορές}} \| \underbrace{11010}_{n=26}}^{64 \text{ bits}}.$$

Τμηματοποίηση Μηνύματος Το νέο μήνυμα $M_{\text{νέο}}$, χωρίζεται σε $N = (n + 1 + k + 64)/512$ τμήματα $M^{(i)}$, $i = 1, \ldots, N$ των 512 bits, έτσι ώστε:

$$M_{\text{νέο}} = M^{(1)} \Big\| \ldots \Big\| M^{(N)}.$$

Κάθε τμήμα $M^{(i)}$, χωρίζεται σε 16 υποτμήματα $M_t^{(i)}$, $t = 0, \ldots, 15$ των 32 bits, έτσι ώστε:

$$M^{(i)} = M_0^{(i)} \Big\| \ldots \Big\| M_{15}^{(i)},$$

για κάθε $i = 1, \ldots, N$.

Αλγόριθμος Κατακερματισμού

Στον επόμενο αλγόριθμο, για κάθε ένα από τα τμήματα $M^{(i)}, i = 1, \ldots, N$, επεξεργάζονται διαδοχικά τα υποτμήματα $M_t^{(i)}, t = 0, \ldots, 15$. Στο τέλος όλων των υπολογιστικών βημάτων, προκύπτει το κατακερματισμένο μήνυμα H των 160 bits.

Αλγόριθμος 5.1. Κατακερματισμός SHA-1.

Είσοδος: Μήνυμα $M = (m_1 \ldots m_n)_2$.
Έξοδος: Κατακερματισμένο μήνυμα $H = (h_1 \ldots h_{160})_2$.

1. Όσο $n \geq 2^{64}$ δώσε νέα είσοδο M.
2. $O \leftarrow O_1 \ldots O_{l_2(n)}$ της $2-$αδικής μορφής του n.
3. $P \leftarrow \underbrace{0 \ldots 0}_{64 - l_2(n) \text{ φορές}} \| O$.
4. $k \leftarrow x$ έτσι ώστε $x = \min \{y \in \mathbb{N} : y + n \equiv 447 \bmod 512\}$.
5. $N \leftarrow (n + 1 + k + 64)/512$.
6. $M \leftarrow M \| 1 \| \underbrace{0 \ldots 0}_{k \text{ φορές}} \| P = M^{(1)} \ldots M^{(N)}$ με

$$\begin{array}{lcl} M^{(1)} & = & M_0^{(1)} \ldots M_{15}^{(1)} \\ \vdots & \vdots & \vdots \\ M^{(N)} & = & M_0^{(N)} \ldots M_{15}^{(N)} \end{array}$$

   όπου το μήκος κάθε $M_t^{(i)}$, $i = 1, \ldots, N$, $t = 0, \ldots, 15$ είναι 32 bits.

7. Για $i = 1$ μέχρι N με βήμα 1
   (α′) Για $t = 0$ μέχρι 15 με βήμα 1
      i. $W_t \leftarrow M_t^i$.
   (β′) Για $t = 16$ μέχρι 79 με βήμα 1
      i. $W_t \leftarrow ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$.
   (γ′) $a \leftarrow H_0^{i-1}$.
   (δ′) $b \leftarrow H_1^{i-1}$.
   (ε′) $c \leftarrow H_2^{i-1}$.

(Δ΄) $d \leftarrow H_3^{i-1}$.

(ζ΄) $e \leftarrow H_4^{i-1}$.

(η΄) Για $t = 0$ μέχρι 79 με βήμα 1

i. $T \leftarrow ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$,

ii. $e \leftarrow d$,

iii. $d \leftarrow c$,

iv. $c \leftarrow ROTL^{30}(b)$,

v. $b \leftarrow a$,

vi. $a \leftarrow T$.

(θ΄) $H_0^{(i)} \leftarrow a + H_0^{(i-1)}$.

(ι΄) $H_1^{(i)} \leftarrow b + H_1^{(i-1)}$.

(ια΄) $H_2^{(i)} \leftarrow b + H_2^{(i-1)}$.

(ιβ΄) $H_3^{(i)} \leftarrow b + H_3^{(i-1)}$.

(ιγ΄) $H_4^{(i)} \leftarrow b + H_4^{(i-1)}$.

8. Επίστρεψε $H_0^{(N)} \Big\| H_1^{(N)} \Big\| H_2^{(N)} \Big\| H_3^{(N)} \Big\| H_4^{(N)}$.

Παράδειγμα 5.10. Έστω ότι έχουμε το μήνυμα:

have a nice day

Εφαρμόζοντας την συνάρτηση SHA-1 στο μήνυμα αυτό, παίρνουμε τον κατακερματισμό

d76cf6cc9bfcd880395ddea39468bbe327a7f6bd.

Αν τώρα, αντί της λέξης have είχαμε την λέξη Have, τότε ο κατακερματισμός του μηνύματος: Have a nice day, θα έδινε

b6bab9bc8870165ecb6c16e713339e4981432239

ο οποίος είναι πολύ διαφορετικός σε σχέση με τον προηγούμενο.

5.2.2 SHA-2

Στην κατηγορία SHA-2, ανήκουν οι συναρτήσεις κατακερματισμού SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 και SHA-512/256. Οι συναρτήσεις αυτές, χρησιμοποιούν παρόμοιες λειτουργίες, αλλά διαφέρουν μεταξύ τους, κυρίως ως προς το μέγεθος του μηνύματος εισόδου και εξόδου [16]. Εδώ θα περιγράψουμε την λειτουργία της SHA-224, η οποία κατακερματίζει το αρχικό μήνυμα σε ένα μήνυμα 224 bits.

Αρκετοί παράμετροι στην SHA-224 είναι ίδιοι με της SHA-1 γι' αυτό η αναφορά τους παραλείπεται.

Ορισμός Συμβόλων

$ROTR^n : \{0,1\}^{32} \to \{0,1\}^{32}$ δεξιά περιστροφή με τύπο:

$$x \mapsto ROTR^n(x) = (x \gg n) \vee (x \ll 32 - n) \tag{5.9}$$

όπου $0 \leq n < 32$.

$SHR^n : \{0,1\}^{32} \to \{0,1\}^{32}$ δεξιά μετατόπιση με τύπο:

$$x \mapsto SHR^n(x) = x \gg n \tag{5.10}$$

όπου $0 \leq n < 32$.

Στον επόμενο πίνακα, δίνονται ορισμένα παραδείγματα πράξεων με τα παραπάνω σύμβολα.

Παράδειγμα 5.11.

Σύμβολο	Πράξη	Αποτέλεσμα
$ROTR^2$	$(\text{5c2f3a17} \gg 2) \vee (\text{5c2f3a17} \ll 30)$	d70bce85
SHR^2	$\text{5c2f3a17} \gg 2$	170bce85

Ορισμός Λογικών Συναρτήσεων

$\sum_0^{\{224\}} : \{0,1\}^{32} \to \{0,1\}^{32}$ με τύπο:

$$\sum_0^{\{224\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$

$\sum_1^{\{224\}} : \{0,1\}^{32} \to \{0,1\}^{32}$ με τύπο:

$$\sum_1^{\{224\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

$\sigma_0^{\{224\}} : \{0,1\}^{32} \to \{0,1\}^{32}$ με τύπο:

$$\sigma_0^{\{224\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$

$\sigma_1^{\{224\}} : \{0,1\}^{32} \to \{0,1\}^{32}$ με τύπο:

$$\sigma_1^{\{224\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

Ορισμός Σταθερών Συναρτήσεων

$K_0^{\{224\}}, \ldots, K_{63}^{\{224\}}$ με τιμές που αντιστοιχούν, από αριστερά προς τα δεξιά, όπως παρακάτω:

428a2f98	71374491	b5c0fbcf	e9b5dba5	3956c25b	59f111f1
923f82a4	ab1c5ed5	d807aa98	12835b01	243185be	550c7dc3
72be5d74	80deb1fe	9bdc06a7	c19bf174	e49b69c1	efbe4786
0fc19dc6	240ca1cc	2de92c6f	4a7484aa	5cb0a9dc	76f988da
983e5152	a831c66d	b00327c8	bf597fc7	c6e00bf3	d5a79147
06ca6351	14292967	27b70a85	2e1b2138	4d2c6dfc	53380d13
650a7354	766a0abb	81c2c92e	92722c85	a2bfe8a1	a81a664b
c24b8b70	c76c51a3	d192e819	d6990624	f40e3585	106aa070
19a4c116	1e376c08	2748774c	34b0bcb5	391c0cb3	4ed8aa4a
5b9cca4f	682e6ff3	748f82ee	78a5636f	84c87814	8cc70208
90befffa	a4506ceb	bef9a3f7	c67178f2		

$H_0^{(0)}$	=	c1059ed8	$H_4^{(0)}$	=	ffc00b31
$H_1^{(0)}$	=	367cd507	$H_5^{(0)}$	=	68581511
$H_2^{(0)}$	=	3070dd17	$H_6^{(0)}$	=	64f98fa7
$H_3^{(0)}$	=	f70e5939	$H_7^{(0)}$	=	befa4fa4

Αλγόριθμος Κατακερματισμού

Η είσοδος του μηνύματος και η διαμόρφωσή του, ακολουθούν την ίδια ακριβώς διαδικασία με τον SHA-1. Στον επόμενο αλγόριθμο, μετά από όλα τα υπολογιστικά βήματα, προκύπτει το κατακερματισμένο μήνυμα H των 224 bits.

Αλγόριθμος 5.2. Κατακερματισμός SHA-224.

Είσοδος: Μήνυμα $M = (m_1 \ldots m_n)_2$.
Έξοδος: Κατακερματισμένο μήνυμα $H = (h_1 \ldots h_{224})_2$.

1. Όσο $n \geq 2^{64}$ δώσε νέα είσοδο M.
2. $O \leftarrow O_1 \dots O_{l_2(n)}$ της $2-$αδικής μορφής του n.
3. $P \leftarrow \underbrace{0 \dots 0}_{64-l_2(n) \text{ φορές}} \| O$.
4. $k \leftarrow x$ έτσι ώστε $x = \min\{y \in \mathbb{N} : y + n \equiv 447 \bmod 512\}$.
5. $N \leftarrow (n + 1 + k + 64)/512$.
6. $M \leftarrow M\|1\|\underbrace{0 \dots 0}_{k \text{ φορές}}\|P = M^{(1)} \dots M^{(N)}$ με

$$\begin{array}{lcl} M^{(1)} & = & M_0^{(1)} \dots M_{15}^{(1)} \\ \vdots & \vdots & \vdots \\ M^{(N)} & = & M_0^{(N)} \dots M_{15}^{(N)} \end{array}$$

   όπου το μήκος κάθε $M_t^{(i)}$, $i = 1, \dots, N$, $t = 0, \dots, 15$ είναι 32 bits.

7. Για $i = 1$ μέχρι N με βήμα 1
   (α′) Για $t = 0$ μέχρι 15 με βήμα 1
      i. $W_t \leftarrow M_t^i$.
   (β′) Για $t = 16$ μέχρι 63 με βήμα 1
      i. $W_t \leftarrow \sigma_1^{\{224\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{224\}}(W_{t-15}) + W_{t-16}$.
   (γ′) $a \leftarrow H_0^{i-1}$.
   (δ′) $b \leftarrow H_1^{i-1}$.
   (ε′) $c \leftarrow H_2^{i-1}$.
   (Ϛ′) $d \leftarrow H_3^{i-1}$.
   (ζ′) $e \leftarrow H_4^{i-1}$.
   (η′) $f \leftarrow H_5^{i-1}$.
   (θ′) $g \leftarrow H_6^{i-1}$.
   (ι′) $h \leftarrow H_7^{i-1}$.
   (ια′) Για $t = 0$ μέχρι 63 με βήμα 1
      i. $T_1 \leftarrow h + \sum_1^{\{224\}}(e) + Ch(e, f, g) + K_t^{\{224\}} + W_t$,
      ii. $T_2 \leftarrow \sum_0^{\{224\}}(a) + Maj(a, b, c)$,
      iii. $h \leftarrow g$,

iv. $g \leftarrow f$,
v. $f \leftarrow e$,
vi. $e \leftarrow d + T_1$,
vii. $d \leftarrow c$,
viii. $c \leftarrow b$,
ix. $b \leftarrow a$,
x. $a \leftarrow T_1 + T_2$.

(ιβ′) $H_0^{(i)} \leftarrow a + H_0^{(i-1)}$.
(ιγ′) $H_1^{(i)} \leftarrow b + H_1^{(i-1)}$.
(ιδ′) $H_2^{(i)} \leftarrow c + H_2^{(i-1)}$.
(ιε′) $H_3^{(i)} \leftarrow d + H_3^{(i-1)}$.
(ιϜ′) $H_4^{(i)} \leftarrow e + H_4^{(i-1)}$.
(ιζ′) $H_5^{(i)} \leftarrow f + H_5^{(i-1)}$.
(ιη′) $H_6^{(i)} \leftarrow g + H_6^{(i-1)}$.
(ιθ′) $H_7^{(i)} \leftarrow h + H_7^{(i-1)}$.

8. Επίστρεψε $H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$.

Παράδειγμα 5.12. Έστω ότι έχουμε πάλι, το μήνυμα:

have a nice day.

Εφαρμόζοντας την συνάρτηση SHA-224 στο μήνυμα αυτό, παίρνουμε τον κατακερματισμό

d57b59507f479f4f6f7d877a1e981e1cf4b42368b86b96fe985a65bb.

Αν τώρα, αντί της λέξης have είχαμε την λέξη Have, τότε ο κατακερματισμός του μηνύματος: Have a nice day, θα έδινε

fbbd3961e7bad10f7aef9afe603681923fcacf93ce52b1054db376e2

ο οποίος είναι πάλι, πολύ διαφορετικός σε σχέση με τον προηγούμενο.

Κεφάλαιο 6

Υπογραφές DSA και ECDSA

Το 1991 το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) των ΗΠΑ, πρότεινε ένα σχήμα ψηφιακής υπογραφής, που ονόμασε Αλγόριθμο Ψηφιακής Υπογραφής (Digital Signature Algorithm). Ο DSA, αρχικά κατατέθηκε από τον David W. Kravitz, πρώην υπάλληλο της NSA [15], [25] και είναι στην ουσία μια αποτελεσματική παραλλαγή της υπογραφής ElGamal [12].

Σήμερα, ο DSA σε συνδυασμό με την θεωρία των ελλειπτικών καμπυλών, έχει μετατραπεί στον Αλγόριθμο Ψηφιακής Υπογραφής Ελλειπτικής Καμπύλης (Elliptic Curve Digital Signature Algorithm). Ο ECDSA, ακολουθεί παρόμοιο σύστημα με τον DSA χρησιμοποιώντας την ομάδα των ελλειπτικών καμπυλών πάνω σε ένα πεπερασμένο σώμα.

Στο κεφάλαιο αυτό, περιγράφεται η δομή και ο τρόπος λειτουργίας αυτών των δύο σχημάτων ψηφιακής υπογραφής. Στη συνέχεια, δίνονται ορισμένα παραδείγματα εκτέλεσης των παραπάνω αλγορίθμων θεωρητικά και πρακτικά.

6.1 Η Ψηφιακή Υπογραφή DSA

6.1.1 Παραγωγή Κλειδιών

Για την δημιουργία κλειδιών, ακολουθείται η παρακάτω διαδικασία:

1. Αρχικά επιλέγεται ένα από τα παρακάτω ζεύγη ακεραίων:
   - $(L, N) = (1024, 160)$
   - $(L, N) = (2048, 224)$
   - $(L, N) = (2048, 256)$

- $(L, N) = (3072, 256)$

2. Επιλέγονται δύο ακέραιοι p, q τέτοιοι ώστε:
   - $2^{L-1} < p < 2^L$ με $p \in \mathbb{P}$.
   - $2^{N-1} < q < 2^N$ όπου q ένας πρώτος διαιρέτης του $p - 1$.
3. Επιλέγεται ένας ακέραιος γ, τέτοιος ώστε να αποτελεί μια αρχική ρίζα κατά μέτρο p.
4. Υπολογίζεται το στοιχείο $g = \gamma^{\frac{p-1}{q}} \bmod p$ του $\mathbb{Z}_p^*$.
5. Επιλέγεται με τυχαίο τρόπο ένας ακέραιος $x \in \{1, \ldots, q-1\}$.
6. Υπολογίζεται ο ακέραιος $y = g^x \bmod p$.

Το αποτέλεσμα με τα ζητούμενα κλειδιά είναι:

Δημόσιο Κλειδί:	(p, q, g, y)
Ιδιωτικό Κλειδί:	x

Παράδειγμα 6.1. Για λόγους απλότητας, στο παράδειγμα αυτό, χρησιμοποιούμε $(L, N) = (12, 8)$.

1. Επιλέγουμε $p = 2467, q = 137$.Πράγματι $2467, 137 \in \mathbb{P}$ και $137 \mid 2467 - 1$ αφού $2466 = 137 \cdot 18$.
2. (α′) Η πρωτογενής ανάλυση του 2466 είναι $2 \cdot 3^2 \cdot 137$.
   (β′) Επιλέγουμε $\gamma = 3$.
   (γ′) Το 3 είναι μια αρχική ρίζα κατά μέτρο 2467 αφού:

$$\begin{aligned} 3^{2466/2} = 3^{1233} &\equiv 2466 \bmod 2467 \\ 3^{2466/3} = 3^{822} &\equiv 2250 \bmod 2467 \\ 3^{2466/137} = 3^{18} &\equiv 342 \bmod 2467 \end{aligned}$$

3. Υπολογίζουμε $g = 3^{2466/137} \bmod 2467 = 342$.
4. Επιλέγουμε $x = 7$.
5. Υπολογίζουμε $y = 342^7 \bmod 2467 = 282$.

Τα ζητούμενα κλειδιά είναι:

Δημόσιο Κλειδί:	$(2467, 137, 342, 282)$
Ιδιωτικό Κλειδί:	7

6.1.2 Υπογραφή Μηνύματος

Για την υπογραφή ενός μηνύματος $M = (m_1 \dots m_n)_2$ ακολουθείται η παρακάτω διαδικασία:

1. Επιλέγεται μια συνάρτηση $SHA - i : \{0,1\}^l \longrightarrow \{0,1\}^e$ για την οποία $i \in \{0,1,2,3\}$, $l \geq n$ και $e \geq N$.
2. Επιλέγεται τυχαία ένας ακέραιος $k \in \{1, \dots q-1\}$ (εφήμερο κλειδί) και κρατείται μυστικός.
3. Υπολογίζεται $r = (g^k \bmod p) \bmod q$.
4. Υπολογίζονται k^{-1} και $SHA - i(M)$.
5. Υπολογίζεται ο ακέραιος $\mathcal{H}(M)$, που είναι τα πρώτα N bits από αριστερά του $SHA - i(M)$ (συμπύκνωση).
6. Υπολογίζεται $s = k^{-1}(\mathcal{H}(M) + xr) \bmod q$.
7. Αν $r = 0$ ή $s = 0$ η διαδικασία επαναλαμβάνεται.

Το αποτέλεσμα της υπογραφής του M, είναι το ζεύγος (r, s).

Παράδειγμα 6.2. Θα συνεχίσουμε με τα δεδομένα του προηγούμενου παραδείγματος.

1. Έστω $M = (m_1 \dots m_{58})_2$ ένα μήνυμα. Επιλέγουμε την $SHA - 2$ με $l = 2^{64}$ και $e = 224$. Η συνάρτηση αυτή είναι η $SHA - 224$.
2. Επιλέγουμε $k = 6$.
3. Υπολογίζουμε $r = (342^6 \bmod 2467) \bmod 137 = 953 \bmod 137 = 131$.
4. Υπολογίζουμε $6^{-1} = 23$ και έστω ότι $SHA_{224}(M) = \underbrace{00011100 \| Q}_{224-\text{bits}}$ με $Q \in \{0,1\}^{216}$.
5. Υπολογίζουμε $\mathcal{H}(M) = 00011100$, επομένως $\mathcal{H}(M) = 28$.
6. Υπολογίζουμε $s = 6^{-1}(28 + 7 \cdot 131) \bmod 137 = 23 \cdot 945 \bmod 137 = 89$.

Το αποτέλεσμα της υπογραφής του M, είναι το ζεύγος $(131, 89)$.

6.1.3 Γνησιότητα Υπογεγραμμένου Μηνύματος

Η διαδικασία αποδοχής ή απόρριψης της υπογραφής (r, s) ενός μηνύματος $M = (m_1 \dots m_n)_2$, έχει ως ακολούθως:

1. Ελέγχεται αν $0 < r < q$ και $0 < s < q$. Αν ένα τουλάχιστον από αυτά δεν ισχύει, τότε απορρίπτεται η υπογραφή.
2. Υπολογίζεται $w = s^{-1} \bmod q$ και $SHA - i(M)$.
3. Υπολογίζεται $\mathcal{H}(M)$.
4. Υπολογίζεται $u_1 = \mathcal{H}(M)w \bmod q$.
5. Υπολογίζεται $u_2 = rw \bmod q$.
6. Υπολογίζεται $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.

Η υπογραφή (r, s) θεωρείται γνήσια, αν $v = r$.

Παράδειγμα 6.3. Θα συνεχίσουμε με τα δεδομένα του προηγούμενου παραδείγματος.

1. Ισχύει $0 < 131 < 137$ και $0 < 89 < 137$ επομένως συνεχίζουμε την διαδικασία.
2. Υπολογίζουμε $w = 89^{-1} \bmod 137 = 117$ και $SHA_{224}(M) = \underbrace{00011100 \| Q}_{224-\text{bits}}$ όπου $Q \in \{0,1\}^{216}$.
3. Υπολογίζουμε $\mathcal{H}(M) = 28$.
4. Υπολογίζουμε $u_1 = 28 \cdot 117 \bmod 137 = 125$.
5. Υπολογίζουμε $u_2 = 131 \cdot 117 \bmod 137 = 120$.
6. Υπολογίζουμε $v = (342^{125} 282^{120} \bmod 2467) \bmod 137 = 953 \bmod 137 = 131$.

Παρατηρούμε ότι $v = r = 131$. Επομένως η υπογραφή γίνεται αποδεκτή ως γνήσια.

Πρόταση 6.1. Αν M είναι ένα μήνυμα, τότε η υπογραφή του είναι το ζεύγος (r, s), αν και μόνο αν, $v = r$.

Απόδειξη. Υποθέτουμε ότι (r, s) είναι η υπογραφή του M. Τότε έχουμε:

$$\begin{aligned} v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= (g^{\mathcal{H}(M)w} y^{rw} \bmod p) \bmod q \\ &= ((g^{\mathcal{H}(M)w} g^{xrw}) \bmod p) \bmod q \\ &= (g^{\mathcal{H}(M)w+xrw} \bmod p) \bmod q \\ &= (g^{w(\mathcal{H}(M)+xr)} \bmod p) \bmod q \\ &= (g^{s^{-1}(\mathcal{H}(M)+xr)} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q \\ &= r. \end{aligned}$$

Αντίστροφα, υποθέτουμε ότι $v = r$. Τότε έχουμε:

$$r = (g^{s^{-1}(\mathcal{H}(M)+xr)} \bmod p) \bmod q$$

Θέτουμε

$$k = s^{-1}(\mathcal{H}(M) + xr)$$

και επομένως,

$$r = (g^k \bmod p) \bmod q \quad \text{και} \quad s = k^{-1}(\mathcal{H}(M) + xr) \bmod q.$$

Συνεπώς, το ζεύγος (r, s) είναι η υπογραφή του M. □

6.2 Η Ψηφιακή Υπογραφή ECDSA

6.2.1 Παραγωγή Κλειδιών

1. Επιλέγεται μία ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$, έτσι ώστε:
   - $p \in \mathbb{P}$.
   - $(2^{80} + 1)^2 < p$.
2. Επιλέγεται ένας ακέραιος q τέτοιος ώστε:
   - $q \in \mathbb{P}$.
   - $q \mid |E(\mathbb{Z}_p)|$.
   - $2^{N-1} < q < 2^N$ έτσι ώστε:
     - $N \in [160, 223]$ όταν $|E(\mathbb{Z}_p)| \leq 2^{10} q$.
     - $N \in [224, 255]$ όταν $|E(\mathbb{Z}_p)| \leq 2^{14} q$.

- $N \in [256, 383]$ όταν $|E(\mathbb{Z}_p)| \leq 2^{16}q$.
- $N \in [384, 511]$ όταν $|E(\mathbb{Z}_p)| \leq 2^{24}q$.
- $N \geq 512$ όταν $|E(\mathbb{Z}_p)| \leq 2^{32}q$.

3. Επιλέγεται ένα σημείο $P = (X_P, Y_P) \in E(\mathbb{Z}_p)$ τάξης q.
4. Επιλέγεται ένας ακέραιος $A \in \{0, \ldots, q-1\}$.
5. Υπολογίζεται $Q = (X_Q, Y_Q) = AP$.

Το αποτέλεσμα με τα ζητούμενα κλειδιά είναι:

Δημόσιο Κλειδί: (P, Q)
Ιδιωτικό Κλειδί: A

Παράδειγμα 6.4. Για λόγους απλότητας, στο παράδειγμα αυτό, επιλέγουμε $p, q \in \mathbb{P}$ μικρού μεγέθους.

1. Επιλέγουμε την ελλειπτική καμπύλη $E : y^2 \equiv x^3 + 2 \bmod 7$, όπου $E(\mathbb{Z}_7) = \{(0,3), (0,4), (3,1), (3,6), (5,1), (5,6), (6,1), (6,6)\} \cup \{O_\infty\}$
2. Επιλέγουμε $q = 3$ για τον οποίο $3 \mid |E(\mathbb{Z}_7)| = 9$.
3. Επιλέγουμε $P = (5,1)$ που είναι τάξης 3. Πράγματι: $(5,1) + (5,1) + (5,1) = (5,6) + (5,1) = O_\infty$.
4. Επιλέγουμε $A = 2$.
5. Υπολογίζουμε $Q = 2\,(5,1) = (5,6)$.

Τα ζητούμενα κλειδιά είναι:

Δημόσιο Κλειδί: $((5,1), (5,6))$
Ιδιωτικό Κλειδί: 2

6.2.2 Υπογραφή Μηνύματος

Για την υπογραφή ενός μηνύματος $M = (m_1 \ldots m_n)_2$ ακολουθείται η παρακάτω διαδικασία:

1. Επιλέγεται μια συνάρτηση $SHA - i : \{0,1\}^l \longrightarrow \{0,1\}^e$ για την οποία $i \in \{0,1,2,3\}$, $l \geq n$ και $e \geq N$.

2. Επιλέγεται τυχαία ένας ακέραιος $k \in \{1, \dots q-1\}$ (εφήμερο κλειδί) και κρατείται μυστικός.

3. Υπολογίζεται $kP = (X_{kP}, Y_{kP})$.

4. Υπολογίζεται $r = X_{kP} \bmod q$.

5. Υπολογίζεται k^{-1} και $SHA-i(M)$.

6. Υπολογίζεται ο ακέραιος $\mathcal{H}(M)$, που είναι τα πρώτα N bits από αριστερά του $SHA-i(M)$.

7. Υπολογίζεται $s = k^{-1}(\mathcal{H}(M) + Ar) \bmod q$.

Το αποτέλεσμα της υπογραφής του M, είναι το ζεύγος (r, s).

Παράδειγμα 6.5. Θα συνεχίσουμε με τα δεδομένα του προηγούμενου παραδείγματος.

1. Έστω $M = (m_1 \dots m_{47})_2$ ένα μήνυμα. Επιλέγουμε την $SHA-2$ με $l = 2^{64}$ και $e = 224$. Η συνάρτηση αυτή είναι η $SHA-224$.

2. Επιλέγουμε $k = 2$ και τον κρατάμε μυστικό.

3. Υπολογίζουμε $2P = (5, 6)$.

4. Υπολογίζουμε $r = 5 \bmod 3 = 2$.

5. Υπολογίζουμε $2^{-1} = 4$ και έστω ότι έχουμε $SHA_{224}(M) = \underbrace{01 \| Q}_{224-\text{bits}}$ όπου $Q \in \{0,1\}^{222}$.

6. Υπολογίζουμε $\mathcal{H}(M) = 01$, επομένως $\mathcal{H}(M) = 1$.

7. Υπολογίζουμε $s = 2^{-1}(1 + 2 \cdot 2) \bmod 3 = 20 \bmod 3 = 2$.

Το αποτέλεσμα της υπογραφής του M, είναι το ζεύγος: $(2, 2)$.

6.2.3 Γνησιότητα Υπογεγραμμένου Μηνύματος

Η διαδικασία αποδοχής ή απόρριψης της υπογραφής (r, s) ενός μηνύματος $M = (m_1 \dots m_n)_2$, έχει ώς ακολούθως:

1. Ελέγχεται αν $0 < r < q$ και $0 < s < q$. Αν ένα τουλάχιστον από αυτά δεν ισχύει, τότε απορρίπτεται η υπογραφή.
2. Υπολογίζεται $w = s^{-1} \bmod q$ και $SHA - i(M)$.
3. Υπολογίζεται $\mathcal{H}(M)$.
4. Υπολογίζεται $u_1 = \mathcal{H}(M)w \bmod q$.
5. Υπολογίζεται $u_2 = rw \bmod q$.
6. Υπολογίζεται $v = X_R \bmod q$ όπου $(X_R, Y_R) = u_1 P + u_2 Q$.

Η υπογραφή (r, s) θεωρείται γνήσια, αν $v = r$.

Παράδειγμα 6.6. Θα συνεχίσουμε με τα δεδομένα του προηγούμενου παραδείγματος.

1. Ισχύει $0 < 2 < 3$ και $0 < 2 < 3$ και επομένως συνεχίζουμε την διαδικασία.
2. Υπολογίζουμε $w = 2^{-1} \bmod 3 = 4 \bmod 3 = 1$ και $SHA_{224}(M) = \underbrace{01 \| Q}_{224-\text{bits}}$ με $Q \in \{0,1\}^{222}$.
3. Υπολογίζουμε $\mathcal{H}(M) = 1$.
4. Υπολογίζουμε $u_1 = 1 \cdot 1 \bmod 3 = 1$.
5. Υπολογίζουμε $u_2 = 2 \cdot 1 \bmod 3 = 2$.
6. Υπολογίζουμε $1P + 2Q = (5,1) + 2\,(5,6) = (5,6)$ και $v = 5 \bmod 3 = 2$.

Παρατηρούμε ότι $v = r = 2$. Επομένως, η υπογραφή είναι γνήσια.

Πρόταση 6.2. Αν M είναι ένα μήνυμα, τότε η υπογραφή του είναι το ζεύγος (r, s), αν και μόνο αν, $v = r$.

Απόδειξη. Υποθέτουμε ότι (r, s) είναι η υπογραφή του M. Τότε έχουμε:

$$\begin{aligned} v &= X_R \bmod q \\ &= (u_1 P + u_2 Q)_{x'x} \bmod q \\ &= (\mathcal{H}(M) w P + r w Q)_{x'x} \bmod q \\ &= (w(\mathcal{H}(M) P + rQ))_{x'x} \bmod q \\ &= (s^{-1}(\mathcal{H}(M) P + ArP))_{x'x} \bmod q \\ &= (s^{-1}(\mathcal{H}(M) + Ar) P)_{x'x} \bmod q \\ &= (kP)_{x'x} \bmod q \\ &= X_{kP} \bmod q \\ &= r. \end{aligned}$$

Αντίστροφα, υποθέτουμε ότι $v = r$. Τότε έχουμε:

$$r = (s^{-1}(\mathcal{H}(M) + Ar) P)_{x'x} \bmod \ q.$$

Θέτουμε

$$k = s^{-1}(\mathcal{H}(M) + Ar)$$

και επομένως,

$$r = X_{kP} \bmod q \ \text{ και } \ s = k^{-1}(\mathcal{H}(M) + Ar) \bmod q.$$

Συνεπώς, το ζεύγος (r, s) είναι η υπογραφή του M. □

6.3 Ασφάλεια Σχήματος Ψηφιακής Υπογραφής

6.3.1 Ασφάλεια DSA

Η ασφάλεια του αλγόριθμου ψηφιακής υπογραφής DSA, στηρίζεται στη δυσκολία υπολογισμού του διακριτού λογάριθμου, ο οποίος διατυπώνεται ως εξής: Δεδομένης μιας ομάδας $\mathbb{Z}_p^*$, ενός ακέραιου $g \in \mathbb{Z}_p^*$ με τάξη έναν πρώτο διαιρέτη q του $p-1$ και ενός ακέραιου $y \in \{1, \ldots, p-1\}$, να βρεθεί ένας ακέραιος $c \in \{1, \ldots, q-1\}$, τέτοιος ώστε $g^c = y \bmod p$. Ο ακέραιος c, μπορεί να είναι είτε το ιδιωτικό κλειδί x είτε το εφήμερο κλειδί k.

Η εφαρμογή των αλγορίθμων Shanks και Pollard χρειάζεται περισσότερες από $\sqrt{q}$ πράξεις μέσα στην υποομάδα $< g >$. Καθώς $q > 2^{159}$, ο υπολογισμός του διακριτού λογάριθμου είναι ανέφικτος. Για μεγάλο μέγεθος p (περισσότερο από 1025 bits), ο ποιο αποτελεσματικός αλγόριθμος είναι ο ρ - αλγόριθμος Pollard [35], ο οποίος χρειάζεται περίπου $\sqrt{\pi q/2}$ βήματα.

Ένας άλλος αλγόριθμος, ο αλγόριθμος του Adleman ή αλγόριθμος Λογισμού Δεικτών, όπως ονομάστηκε προηγουμένως, είναι υποεκθετικού χρόνου και κατά συνέπεια ταχύτερος και από τους δύο προηγούμενους. Ωστόσο αυτός, δεν μπορεί να εφαρμοστεί στην υποομάδα $< g >$ αλλά στην ομάδα $\mathbb{Z}_p^*$. Επομένως, καθώς το μήκος του p είναι συνήθως ίσο με 1024, ούτε και αυτός είναι αποτελεσματικός.

Γενικά, δεν υπάρχει γνωστός αλγόριθμος υποεκθετικού χρόνου για τον υπολογισμό του διακριτού λογάριθμου στην υποομάδα $< g >$.

6.3.2 Ασφάλεια ECDSA

Η ασφάλεια του ECDSA, στηρίζεται στο πρόβλημα του διακριτού λογάριθμου στις ελλειπτικές καμπύλες (ECDLP). Η διατύπωση του προβλήματος αυτού είναι η εξής: Δεδομένης μιας ελλειπτικής καμπύλης E επί του $\mathbb{Z}_p$, ενός σημείου $P \in E(\mathbb{Z}_p)$, το οποίο έχει τάξη n και ενός σημείου $Q \in E(\mathbb{Z}_p)$, να βρεθεί ένας ακέραιος l με $0 \leq l \leq n-1$, τέτοιος ώστε $P = lQ$. Ο ακέραιος l, μπορεί να είναι είτε το ιδιωτικό κλειδί A είτε το εφήμερο κλειδί k.

Μια διαφορετική εκδοχή για την ασφάλεια των ελλειπτικών καμπυλών είναι το πλήθος των σημείων της ελλειπτικής καμπύλης. Όσο μεγαλύτερο είναι το πλήθος των σημείων μιας ελλειπτικής καμπύλης, τόσο μεγαλύτερη είναι και η εξαντλητική αναζήτηση που θα πραγματοποιείται στα σημεία της.

Τα τελευταία χρόνια, το ECDLP έχει αποσπάσει το ενδιαφέρον αρκετών κορυφαίων μαθηματικών σε όλο τον κόσμο.

Συναφείς Εργασίες

Ο αλγόριθμος Pohlig and Hellman [34], μετατρέπει τον προσδιορισμό του ακέραιου l, στον προσδιορισμό του ακεραίου $l \bmod x$, όπου x οι πρώτοι διαιρέτες του n. Έτσι, για να επιτευχθεί το μέγιστο δυνατό επίπεδο ασφάλειας, ο αριθμός n πρέπει είναι πρώτος.

Γενικά, ο καλύτερος αλγόριθμος μέχρι σήμερα για το ECDLP είναι ο ρ - αλγόριθμος Pollard, ο οποίος χρειάζεται περίπου $\sqrt{\pi n/2}$ πράξεις μέσα στην ομάδα της ελλειπτικής καμπύλης. Το 1991 οι Paul van Oorschot και Michael Wiener [33], απέδειξαν ότι ο ρ - αλγόριθμος Pollard μπορεί να μετατραπεί σε παράλληλο αλγόριθμο, έτσι ώστε αν αυτός εκτελείται με r επαναλήψεις, οι αναμενόμενες πράξεις μέσα στην ομάδα της ελλειπτικής καμπύλης για κάθε επανάληψη να είναι $(\sqrt{\pi n/2})/r$.

Οι Menezes, Okamoto και Vanstone [26] και Frey και Ruck [17], απέδειξαν πως το ECDLP μπορεί να απλοποιηθεί στο DLP σε μια επέκταση σώματος $\mathbb{Z}_p$, με αλγορίθμους υποεκθετικού χρόνου. Ωστόσο, οι αλγόριθμοι αυτοί είναι αποδοτικοί μόνο για υπεριδιάζουσες καμπύλες (Ορισμός 4.2) και επομένως αυτές δεν παρέχουν επαρκές επίπεδο ασφάλειας.

Επίσης, μια άλλη κατηγορία αλγορίθμων, εφαρμόζονται αποδοτικά σε μη ομαλές καμπύλες (Ορισμός 4.3). Οι αλγόριθμοι αυτοί, ανακαλύφθηκαν ξεχωριστά, από τους Semaev [45], Smart [46], και Satoh και Araki [42], ενώ γενικεύθηκαν από τον Ruck [40]. Οι καμπύλες αυτές όμως, όπως και οι υπεριδιάζουσες, δεν παρέχουν επαρκές επίπεδο ασφάλειας και γι'αυτό η χρήση τους αποφεύγεται.

Μερικές ακόμη εργασίες πάνω στο ECDLP βρίσκονται στην βιβλιογραφία: [1], [4], [9], [29], [47].

6.4 Υλοποίηση ECDSA με το Πρόγραμμα SAGE

Στην ενότητα αυτή, θα δώσουμε χρήσιμες εντολές, για την υλοποίηση της υπογραφής ECDSA, με την χρήση του προγράμματος SAGE [53].

Οι εντολές αυτές, αποτελούν συναρτήσεις πολλών μεταβλητών, για την δημιουργία οποιονδήποτε απαραίτητων στοιχείων επιθυμεί κάθε φορά ο χρήστης. Κάθε συνάρτηση, μπορεί να χρησιμοποιηθεί ανεξάρτητα από τις υπόλοιπες, αλλά και συνδυαστικά, ώστε ο ενδιαφερόμενος χρήστης να μπορεί να κατασκευάσει το δικό του πρόγραμμα. Κατ' αυτήν την έννοια, οι συναρτήσεις αυτές, αποτέλεσαν σημαντικές δομές, με τις οποίες υλοποιήθηκαν οι επιθέσεις που αναλύονται στο Κεφάλαιο 8.

Σε όλες τις συναρτήσεις που ακολουθούν, οι μεταβλητές a,b και p, αποτελούν τους συντελεστές a,b και p αντίστοιχα, της ελλειπτικής καμπύλης $E : y^2 \equiv x^3 + ax + b \bmod p$.

6.4.1 Στοιχειώδεις Συναρτήσεις

- **numbers(n,a,b,x)**:
  - **numbers(n,a,b,0)**: επιστρέφει σε λίστα, n το πλήθος τυχαίους πρώτους αριθμούς, μεγέθους από a bits έως b bits.
  - **numbers(1,0,0,x)**: επιστρέφει το μήκος του αριθμού x (μέγεθος σε bits).

- **elliptic_curve(n,a,b,p,prt)**:

  - **elliptic_curve(1,a,b,p,0)**: επιστρέφει σε λίστα, την ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$ και την τάξη αυτής.
  - **elliptic_curve(n,0,0,p,prt)**: επιστρέφει (prt=0) ή εμφανίζει (prt=1) σε λίστα, n το πλήθος τυχαίες ελλειπτικές καμπύλες E επί του $\mathbb{Z}_p$ με $|E(\mathbb{Z}_p)| \in \mathbb{P}$, με τις αντίστοιχες τάξεις τους.

- **prime_divisors(x,n)**:
  - **prime_divisors(x,0)**: επιστρέφει σε λίστα, όλους τους πρώτους διαιρέτες του x.
  - **prime_divisors(x,1)**: επιστρέφει τον μεγαλύτερο πρώτο διαιρέτη του x.

- **points_generation(n,q,a,b,p)**: επιστρέφει σε λίστα, την ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$, την τάξη αυτής και n το πλήθος τυχαία σημεία της με τάξη q.
  - **points_generation(n,0,0,0,p)**: επιστρέφει σε λίστα, μια τυχαία ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$ με $|E(\mathbb{Z}_p)| \in \mathbb{P}$, την τάξη αυτής και n το πλήθος τυχαία σημεία της, με τάξη $|E(\mathbb{Z}_p)|$.

- **keys_generation(q,a,b,p,Xp,Yp,A)**: επιστρέφει σε λίστα, την ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$, την τάξη αυτής και τα κλειδιά $P = (Xp, Yp)$, Q, A, όπου P είναι τάξης q.
  - **keys_generation(q,a,b,p,Xp,Yp,0)**: επιστρέφει σε λίστα, την ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$, την τάξη αυτής και τα κλειδιά $P = (Xp, Yp)$, Q, A, όπου P είναι τάξης q και A τυχαία επιλεγμένο.
  - **keys_generation(0,0,0,p,0,0,A)**: επιστρέφει σε λίστα, μια τυχαία ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$ με $|E(\mathbb{Z}_p)| \in \mathbb{P}$, την τάξη αυτής, τυχαία κλειδιά P και Q και το κλειδί A.
  - **keys_generation(0,0,0,p,0,0,0)**: επιστρέφει σε λίστα, μια τυχαία ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$ με $|E(\mathbb{Z}_p)| \in \mathbb{P}$, την τάξη αυτής και τυχαία κλειδιά P, Q, A.

- **message_representation(prt)**: επιστρέφει (prt=0) ή εμφανίζει (prt=1) σε λίστα, την αναπαράσταση ενός μηνύματος:
  - στην $2-$αδική του μορφή (κωδικοποίηση ascii),

  - στην $10-$αδική μορφή της $2-$αδικής του μορφής (κωδικοποίηση ascii),
  - στην $16-$αδική μορφή της $2-$αδικής του μορφής (κωδικοποίηση ascii).

  και τον κατακερματισμό SHA-224 του μηνύματος:

  - στην $2-$ αδική του μορφή,
  - στην $10-$ αδική του μορφή,
  - στην $16-$ αδική του μορφή.

- **ECDSA_Signature(q,a,b,p,Xp,Yp,A,k,prt,H)**:
  - Στις επόμενες τρείς συναρτήσεις, αν A,k=0, τότε αυτές λαμβάνουν αυθαίρετες τιμές.
  - **ECDSA_Signature(q,a,b,p,Xp,Yp,A,k,prt,0)**: επιστρέφει (prt=0) ή εμφανίζει (prt=1) σε λίστα, την αναπαράσταση του μηνύματος στην $10-$ αδική μορφή, την ελλειπτική καμπύλη E επί του $\mathbb{Z}_p$, την τάξη αυτής, τον αριθμό q, το δημόσιο κλειδί P, το δημόσιο κλειδί Q, το ιδιωτικό κλειδί A, το εφήμερο κλειδί k, τον ακέραιο k^{-1}, τον ακέραιο A^{-1}, την συμπύκνωση

    $$\mathcal{H}(M) = \left\lfloor SHA_{224}(M) \cdot 2^{l_2(q)-224} \right\rfloor$$

    και την υπογραφή (r, s) του μηνύματος.
  - **ECDSA_Signature(q,a,b,p,Xp,Yp,A,k,prt,H)**: επιστρέφει (prt=0) ή εμφανίζει (prt=1) την προηγούμενη λίστα, όπου H η τιμή $\mathcal{H}(M)$ της συμπύκνωσης του μηνύματος.
  - **ECDSA_Signature(0,0,0,p,0,0,A,k,prt,H)**: επιστρέφει (prt=0) ή εμφανίζει (prt=1) κατά περίπτωση, τα προηγούμενα αποτελέσματα, με τη διαφορά ότι η ελλειπτική καμπύλη είναι τυχαία επιλεγμένη.

- **Message_Authenticity(q,a,b,p,Xp,Yp,Xq,Yq,r,s,H)**: επιστρέφει μήνυμα, για το αν η υπογραφή (r, s) με τις τιμές των μεταβλητών που χρησιμοποιήθηκαν είναι γνήσια ή όχι. Αν H=0 το κείμενο εισάγεται από τον χρήστη ενώ αν H$\neq$0 τότε εισάγεται $H = \mathcal{H}(M)$.

Στη συνέχεια, παρουσιάζουμε ένα παράδειγμα υπογραφής ενός αυθαίρετου μηνύματος, με την χρήση των παραπάνω συναρτήσεων.

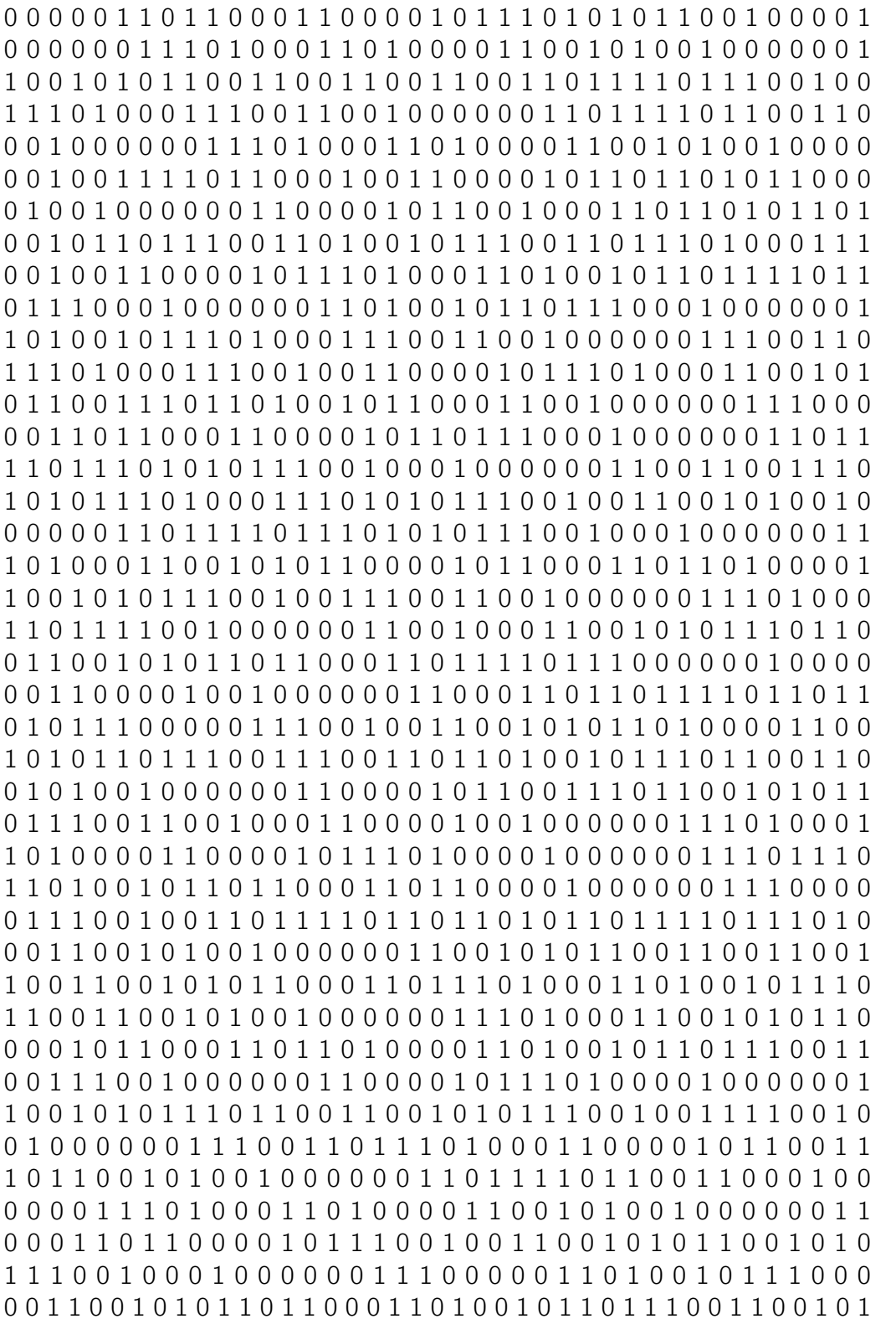
6.4.2 Εφαρμογή σε Κείμενο

Παράδειγμα 6.7. Έστω ότι θέλουμε να υπογράψουμε το παρακάτω μήνυμα M:

We have to raise the bar for teacher preparation so that excellent programs and practices are the norm across our nation we applaud the efforts of the Obama administration in its strategic plan our future our teachers to develop a comprehensive agenda that will promote effective teaching at every stage of the career pipeline

Χρησιμοποιώντας την εντολή message_representation(1) και εισάγωντας το παραπάνω μήνυμα, η $2-$αδική μορφή (κωδικοποίηση ascii) του μηνύματος είναι:

0 1 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
0 0 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0
1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 0 1 1
1 0 0 1 1 0 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1
1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0
0 1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0
0 1 1 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 1 0 1 1 0 1 0 0
0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0
1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 1 0 0
0 0 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0
0 0 0 0 0 1 1 1 0 0 1 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1
0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1
1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 0
1 1 0 0 1 0 1 0 1 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0
0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0
1 0 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1
1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0
0 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 1 1 0
0 1 0 1 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 1
0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 1 0 0
1 0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1 0
0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 1
0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0
1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 1 0 1 1 1 0 1
0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0
1 1 1 0 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1



η $16-$ αδική του μορφή (κωδικοποίηση ascii) είναι:

5 7 6 5 2 0 6 8 6 1 7 6 6 5 2 0 7 4 6 f 2 0 7 2 6 1 6 9 7 3 6 5 2 0 7 4 6
8 6 5 2 0 6 2 6 1 7 2 2 0 6 6 6 f 7 2 2 0 7 4 6 5 6 1 6 3 6 8 6 5 7 2 2 0 7 0
7 2 6 5 7 0 6 1 7 2 6 1 7 4 6 9 6 f 6 e 2 0 7 3 6 f 2 0 7 4 6 8 6 1 7 4 2 0 6
5 7 8 6 3 6 5 6 c 6 c 6 5 6 e 7 4 2 0 7 0 7 2 6 f 6 7 7 2 6 1 6 d 7 3 2 0 6 1
6 e 6 4 2 0 7 0 7 2 6 1 6 3 7 4 6 9 6 3 6 5 7 3 2 0 6 1 7 2 6 5 2 0 7 4 6 8
6 5 2 0 6 e 6 f 7 2 6 d 2 0 6 1 6 3 7 2 6 f 7 3 7 3 2 0 6 f 7 5 7 2 2 0 6 e
6 1 7 4 6 9 6 f 6 e 2 0 7 7 6 5 2 0 6 1 7 0 7 0 6 c 6 1 7 5 6 4 2 0 7 4 6 8
6 5 2 0 6 5 6 6 6 6 6 f 7 2 7 4 7 3 2 0 6 f 6 6 2 0 7 4 6 8 6 5 2 0 4 f 6 2
6 1 6 d 6 1 2 0 6 1 6 4 6 d 6 9 6 e 6 9 7 3 7 4 7 2 6 1 7 4 6 9 6 f 6 e 2 0
6 9 6 e 2 0 6 9 7 4 7 3 2 0 7 3 7 4 7 2 6 1 7 4 6 5 6 7 6 9 6 3 2 0 7 0 6 c
6 1 6 e 2 0 6 f 7 5 7 2 2 0 6 6 7 5 7 4 7 5 7 2 6 5 2 0 6 f 7 5 7 2 2 0 7 4
6 5 6 1 6 3 6 8 6 5 7 2 7 3 2 0 7 4 6 f 2 0 6 4 6 5 7 6 6 5 6 c 6 f 7 0 2 0
6 1 2 0 6 3 6 f 6 d 7 0 7 2 6 5 6 8 6 5 6 e 7 3 6 9 7 6 6 5 2 0 6 1 6 7 6 5
6 e 6 4 6 1 2 0 7 4 6 8 6 1 7 4 2 0 7 7 6 9 6 c 6 c 2 0 7 0 7 2 6 f 6 d 6 f
7 4 6 5 2 0 6 5 6 6 6 6 6 5 6 3 7 4 6 9 7 6 6 5 2 0 7 4 6 5 6 1 6 3 6 8 6 9
6 e 6 7 2 0 6 1 7 4 2 0 6 5 7 6 6 5 7 2 7 9 2 0 7 3 7 4 6 1 6 7 6 5 2 0 6 f
6 6 2 0 7 4 6 8 6 5 2 0 6 3 6 1 7 2 6 5 6 5 7 2 2 0 7 0 6 9 7 0 6 5 6 c 6 9 6 e 6 5

και η $10-$ αδική του μορφή (κωδικοποίηση ascii) είναι:

$$
\begin{aligned}
M \;=\; & 41636617521644834637687190668413111637197108378388782015657534535951866377748079916509980326857763717622870102203447\\
& 71351681133750378305913933894499787590666213392904000982866363723241275973840702130819347811304415293103326743682036\\
& 32276725051262374684825518958600329286563653640142821975188626885714961106845401683353257538232622358583744485251805\\
& 62576648200693189357531952214000447652003987995203039617440531613990131748906229344159791209161525118747243602974739\\
& 89047191111258669778760818828287282120225238187687570398164755394492745219306607081556948927606139629152084972240918\\
& 05049270624898135203220589879569934152671298371401906945803526773007790007829875593126730574510698898876883040498944\\
& 24005859453223396292888157460460533052394783989961218809353084497319472484134256019861093.
\end{aligned}
$$

Ο κατακερματισμός $SHA_{224}(M)$ του μηνύματος στην $2-$ αδική του μορφή είναι:

0 1 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1 0 1 1 1 1
1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0
0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1
0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0
0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 1
0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 1 0 1 1 0 0 1

στην $16-$αδική του μορφή είναι:

7b906a35ff984db1184a14c35feb60061f248dcbcc8c4e9c1e4c6459

και στην $10-$αδική του μορφή είναι:

$$\begin{aligned} SHA_{224}(M) &= 130128207136823503921405694298994 \\ &\quad 9810370414418130128994003765509436 1. \end{aligned}$$

Χρησιμοποιώντας την εντολή numbers(1,147,147,0) παίρνουμε τον πρώτο

$$p = 2^{147} + 51.$$

Χρησιμοποιώντας την εντολή elliptic_curve(1,0,0,2^147+51,0) παίρνουμε την ελλειπτική καμπύλη

$$E : y^2 \equiv x^3 + \underbrace{182}_{a} x + \underbrace{12430701893}_{b} \bmod \left(2^{147} + 51\right)$$

με

$$|E(\mathbb{Z}_p)| = 2^{147} + 5073516209418336782121.$$

Επιλέγουμε

$$q = 2^{147} + 507351620941833678212 1$$

και εκτελώντας την εντολή points_generation(1,q,a,b,p) με τις τιμές των μεταβλητών που ορίστηκαν, παίρνουμε το σημείο

$$\begin{aligned} P &= (155478435296114276577582312035537317478986353, \\ &\quad 879956048214914285355551660051243492303753 49) \\ &= (X_p, Y_p) \end{aligned}$$

τάξης q. Εκτελώντας την εντολή keys_generation(q,a,b,p,Xp,Yp,0) παίρνουμε το δημόσιο κλειδί P,

$$\begin{aligned} Q &= (135195924388008732217191226858593035295968639, \\ &\quad 114625461726350336523129290769990662011144010) \\ &= (X_q, Y_q) \end{aligned}$$

και το ιδιωτικό κλειδί

$$A = 136976179271093535520703867126833946623683366.$$

Εκτελώντας την εντολή ECDSA_Signature(q,a,b,p,Xp,Yp,A,0,1,0) με τις τιμές των μεταβλητών που ορίστηκαν και εισάγοντας σε αυτή το μήνυμα, παίρνουμε το εφήμερο κλειδί και την συμπύκνωση του μηνύματος, αντίστοιχα

$$\begin{aligned} k &= 95999632649419840363885374285065064090070908 \\ \mathcal{H}(M) &= 17222324814378388528313517212147804914738790 4 \end{aligned}$$

και την υπογραφή του μηνύματος

$$\begin{aligned} (r,s) = (&163673129307877977960716238707085664553273219, \\ &662617127485909573656284197827903093570280). \end{aligned}$$

Εδώ σημειώνουμε, ότι το ιδιωτικό και το εφήμερο κλειδί A και k αντίστοιχα, έχουν και τα δύο μήκος 147 bits, το οποίο διαπιστώνεται με τις εντολές numbers(1,0,0,A) και numbers(1,0,0,k), αντίστοιχα.

Τέλος, για να επαληθεύσουμε την γνησιότητα του μηνύματος, εισάγουμε στην συνάρτηση Message_Authenticity(q,a,b,p,Xp,Yp,Xq,Yq,r,s,0) το μήνυμα M και τις τιμές όλων των μεταβλητών, από την οποία λαμβάνουμε

$$v = r$$

και έτσι το αρχικό μας μήνυμα M είναι αυθεντικό.

Το παράδειγμα αυτό, θα μπορούσε εξαρχής να υλοποιηθεί, με την εκτέλεση απλά της εντολής ECDSA_Signature(0,0,0,2^147+51,0,0,0,0,1,0).

Κεφάλαιο 7

Δικτυωτά

Η μελέτη των δικτυωτών, ιστορικά ξεκίνησε από το ενδιαφέρον που επέδειξαν γνωστοί μαθηματικοί όπως ο Lagrange, ο Gauss και ο Minkowski. Η ανάπτυξη της θεωρίας των δικτυωτών από τότε, έχει λάβει σημαντικές διαστάσεις και αποτελεί σύγχρονο αντικείμενο έρευνας της επιστήμης των υπολογιστών, της κρυπτογραφίας, της ασφάλειας του κυβερνοχώρου κ.α.

Τα δικτυωτά αποτελούν στην πράξη γεωμετρικά αντικείμενα τα οποία παρουσιάζουν σημαντικές αλγεβρικές ιδιότητες, μερικές εκ' των οποίων θα αναλύσουμε στη συνέχεια. Στις μέρες μας είναι τόσο χρήσιμα, αφού μπορούν να χρησιμοποιηθούν, σαν εργαλεία επίλυσης μεγάλου συνόλου προβλημάτων.

7.1 Βασικές Έννοιες

Ας είναι $\vec{x_1}, \ldots, \vec{x_n}$ διανύσματα του $m-$διάστατου χώρου $\mathbb{R}^m$, με $m, n > 1$.

Ορισμός 7.1. Ονομάζουμε *γραμμικό συνδυασμό* των $\vec{x_i}$, $i = 1, \ldots, n$ οποιοδήποτε διάνυσμα $\vec{x}$ με

$$\vec{x} = k_1\vec{x_1} + \cdots + k_n\vec{x_n}$$

όπου $k_i \in \mathbb{Z}$, $i = 1, \ldots, n$.

Ορισμός 7.2. Τα διανύσματα $\vec{x_i}$, $i = 1, \ldots, n$ ονομάζονται *γραμμικά ανεξάρτητα*, αν για κάθε $k_i \in \mathbb{Z}$, $i = 1, \ldots, n$ ισχύει η συνεπαγωγή:

$$k_1\vec{x_1} + \cdots + k_n\vec{x_n} = \vec{0} \quad \Rightarrow \quad k_1 = \cdots = k_n = 0$$

Ορισμός 7.3. Αν τα διανύσματα $\vec{x_i}$, $i = 1, \ldots, n$ είναι γραμμικά ανεξάρτητα, τότε το σύνολο όλων των δυνατών γραμμικών συνδυασμών τους, ονομάζεται *δικτυωτό* και συμβολίζεται με

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} k_i \vec{x_i} : k_i \in \mathbb{Z} \right\}.$$

Η ακολουθία $\mathcal{B} : \vec{x_1}, \ldots, \vec{x_n}$, ονομάζεται *βάση* του δικτυωτού $\mathcal{L}$ και λέμε ότι *παράγει* το δικτυωτό $\mathcal{L}$, το οποίο συμβολίζουμε με

$$\mathcal{L} = \mathcal{L}(\vec{x_1}, \ldots, \vec{x_n}).$$

Ο αριθμός m ονομάζεται *διάσταση* του δικτυωτού $\mathcal{L}$ και ο αριθμός n *βαθμός* του δικτυωτού $\mathcal{L}$. Αν $m = n$ το δικτυωτό $\mathcal{L}$ ονομάζεται *πλήρες δικτυωτό*.

Αρκετές φορές, λέμε ότι το δικτυωτό $\mathcal{L}$, παράγεται από τις γραμμές του πίνακα

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{pmatrix},$$

όπου $x_{i,j}$ οι συντελεστές των διανυσμάτων $\vec{x_i}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. Τότε έχουμε:

$$\mathcal{L} = \left\{ \left(\sum_{i=1}^{n} k_i x_{i,1}, \sum_{i=1}^{n} k_i x_{i,2}, \ldots, \sum_{i=1}^{n} k_i x_{i,m} \right) : (k_1, k_2, \ldots, k_n) \in \mathbb{Z}^n \right\}.$$

Στη περίπτωση αυτή, το δικτυωτό $\mathcal{L}$ συμβολίζεται απλά με $\mathcal{L}(X)$ όπου

$$\mathcal{L}(X) = \{Xk : k \in \mathbb{Z}^n\}.$$

Με βάση τον προηγούμενο ορισμό, λέμε ότι το $span$ ενός δικτυωτού $\mathcal{L}(X)$, είναι το σύνολο

$$span\left(\mathcal{L}(X)\right) = \{Xk : k \in \mathbb{R}^n\}.$$

Επίσης, ως *ορίζουσα* ενός δικτυωτού $\mathcal{L}(X)$ θα εννοούμε την ποσότητα

$$vol(\mathcal{L}(X)) = \sqrt{\det(X^T X)}$$

ενώ με $\det(\mathcal{L})$ θα συμβολίζουμε την τιμή $\det(X)$.

Ορισμός 7.4. Έστω ένα διάνυσμα $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Ονομάζουμε *Ευκλείδεια νόρμα* του διανύσματος $\vec{x}$ και το συμβολίζουμε με $\|\vec{x}\|$, την ποσότητα

$$\|\vec{x}\| = (x_1^2 + \cdots + x_n^2)^{1/2}$$

η οποία είναι η απόσταση του διανύσματος από την αρχή των αξόνων, την οποία ονομάζουμε και *μήκος* του διανύσματος $\vec{x}$.

Ορισμός 7.5. Έστω $\mathcal{B} : \vec{x_1}, \ldots, \vec{x_n}$ μια βάση ενός δικτυωτού $\mathcal{L}$ σε έναν $m-$διάστατο χώρο $V \subseteq \mathbb{R}^m$. Η βάση αυτή, ονομάζεται *ορθογώνια βάση* του V, αν

$$\vec{x_i} \cdot \vec{x_j} = \sum_{k=1}^{m} x_{i,k} x_{j,k} = 0 \ , \ \forall \ i \neq j$$

ενώ αν $\|\vec{x_i}\| = 1$, $i = 1, \ldots, n$ ονομάζεται *ορθοκανονική βάση* του V.

Ορισμός 7.6. Έστω $\mathcal{L}$ ένα δικτυωτό. Ονομάζουμε *ελάχιστη απόσταση* του $\mathcal{L}$ και την συμβολίζουμε με $\lambda(\mathcal{L})$, την μικρότερη απόσταση μεταξύ οποιουδήποτε ζεύγους διανυσμάτων $\vec{x}, \vec{y}$ του $\mathcal{L}$. Δηλαδή,

$$\lambda(\mathcal{L}) = \inf \{\|\vec{x} - \vec{y}\| : \vec{x}, \vec{y} \in \mathcal{L}, \vec{x} \neq \vec{y}\} .$$

Με $\inf S$ συμβολίζεται το μεγαλύτερο κάτω φράγμα ενός συνόλου S.

Ορισμός 7.7. Έστω $\mathcal{L}$ ένα δικτυωτό βαθμού n. Για κάθε $i \in \{1, \ldots, n\}$ ορίζουμε το $i-$οστό *διαδοχικό ελάχιστο* που είναι ο αριθμός

$$\lambda_i(\mathcal{L}) = \inf \left\{ \dim \left(\left(\mathcal{L} \cap \bar{B}(0, r) \right) \right) \geq i \right\}$$

με

$$\bar{B}(0, r) = \{\vec{x} \in \mathbb{R}^m : \|\vec{x}\| \leq r\}$$

όπου dim η διάσταση του γραμμικού χώρου $\left(\mathcal{L} \cap \bar{B}(0, r) \right)$.

Ορισμός 7.8. Έστω $\mathcal{L}$ ένα δικτυωτό διάστασης n. Η ποσότητα

$$Gauss(\mathcal{L}) = \sqrt{\frac{n}{2\pi e}} (\det(\mathcal{L}))^{1/n}$$

καλείται *αναμενόμενο μικρότερο μήκος κατά Gauss*.

7.2 Θεμελιώδεις Προβλήματα και Θεωρήματα

Τα κλασικά υπολογιστικά προβλήματα που συνδέονται με δικτυωτά και εκμεταλλεύεται η κρυπτογραφία σήμερα είναι τα εξής:

1. Πρόβλημα μικρότερου διανύσματος (Shortest Vector Problem - SVP).
2. Πρόβλημα εγγύτερου διανύσματος (Closest Vector Problem - CVP).

Υπάρχουν πολλές σημαντικές παραλλαγές των SVP και CVP που προκύπτουν τόσο στη θεωρία όσο και στην πράξη. Μερικά μόνο από τα προβλήματα αυτά, είναι τα εξής:

1. Πρόβλημα μικρότερης βάσης (Shortest Basis Problem - SBP).
2. Κατά προσέγγιση πρόβλημα μικρότερου διανύσματος (Approximate Shortest Vector Problem - apprSVP).
3. Κατά προσέγγιση πρόβλημα εγγύτερου διανύσματος (Approximate Closest Vector Problem - apprCVP).

Η περιγραφή των προηγούμενων προβλημάτων έχει ως εξής:

SVP: Να βρεθεί ένα μικρότερο μη μηδενικό διάνυσμα σε ένα δικτυωτό. Δηλαδή, να βρεθεί ένα μη μηδενικό διάνυσμα $\vec{v} \in \mathcal{L}$ τέτοιο ώστε $\|\vec{v}\| \leq \|\vec{x}\|$ για κάθε άλλο διάνυσμα $\vec{x} \in \mathcal{L}$.

CVP: Έχοντας ένα διάνυσμα $\vec{w} \in \mathbb{R}^m$ που να μην ανήκει σε ένα δικτυωτό $\mathcal{L}$, να βρεθεί ένα διάνυσμα $\vec{v} \in \mathcal{L}$ το οποίο να βρίσκεται εγγύτερα στο $\vec{w}$. Δηλαδή, να βρεθεί ένα διάνυσμα $\vec{v} \in \mathcal{L}$ τέτοιο ώστε $\|\vec{v} - \vec{w}\| \leq \|\vec{x} - \vec{w}\|$ για κάθε άλλο διάνυσμα $\vec{x} \in \mathcal{L}$.

SBP: Να βρεθεί μια βάση $\mathcal{B} : \vec{v_1}, \ldots, \vec{v_n}$ ενός δικτυωτού που να είναι η μικρότερη σε κάποια περίπτωση, όπως για παράδειγμα στην ελαχιστοποίηση της ποσότητας

$$\max_{1 \leq i \leq n} \|\vec{v_i}\| \quad \text{ή} \quad \sum_{i=1}^{n} \|\vec{v_i}\|^2 .$$

Υπάρχουν πολλές διαφορετικές εκδοχές του SBP, ανάλογα με το πως κάποιος αποφασίζει να μετρήσει το "μέγεθος" της βάσης.

apprSVP: Έστω $\psi(n)$ μια συνάρτηση του n. Να βρεθεί ένα μη μηδενικό διάνυσμα ενός δικτυωτού $\mathcal{L}$ διάστασης n, το οποίο να μην είναι μεγαλύτερο από το $\psi(n)$ − πλάσιο ενός μικρότερου διανύσματος του δικτυωτού. Με άλλα λόγια, αν $\vec{v_s}$ είναι ένα μικρότερο διάνυσμα ενός δικτυωτού $\mathcal{L}$ διάστασης n, να βρεθεί ένα μη μηδενικό διάνυσμα $\vec{v} \in \mathcal{L}$, τέτοιο ώστε

$$\|\vec{v}\| \leq \psi(n) \|\vec{v_s}\| .$$

apprCVP: Το πρόβλημα αυτό είναι παρόμοιο με το apprSVP, αλλά τώρα ψάχνουμε για ένα διάνυσμα που να είναι μια κατά προσέγγιση λύση για το CVP, αντί για μια κατά προσέγγιση λύση για το SVP. Δηλαδή, ψάχνουμε ένα μη μηδενικό διάνυσμα ενός δικτυωτού $\mathcal{L}$ διάστασης n, το οποίο να μην είναι μεγαλύτερο από το $\psi(n)$ − πλάσιο ενός εγγύτερου διανύσματος του δικτυωτού.

Μέχρι σήμερα, αν και δεν έχουν βρεθεί ακριβείς αλγόριθμοι που να επιλύουν τα παραπάνω προβλήματα σε πολυωνυμική πολυπλοκότητα χρόνου, έχουν αναπτυχθεί αρκετά καλοί προσεγγιστικοί αλγόριθμοι πολυωνυμικού χρόνου, όπως ο LLL (Lenstra–Lenstra–Lovasz) και ο αλγόριθμος του Babai, αντίστοιχα [3]. Ο γρηγορότερος ακριβής αλγόριθμος είναι αυτός των Micciancio και Βούλγαρη (2010) [28] ο οποίος έχει εκθετική πολυπλοκότητα χρόνου.

Θεώρημα 7.1. Κάθε δικτυωτό $\mathcal{L}$ διάστασης n περιέχει ένα μη μηδενικό διάνυσμα $\vec{v} \in \mathcal{L}$ τέτοιο ώστε

$$\|\vec{v}\| \leq \sqrt{n} \det(\mathcal{L})^{1/n} .$$

Το προηγούμενο θεώρημα είναι γνωστό ως *Θεώρημα του Hermite*.

Θεώρημα 7.2. Έστω $\mathcal{L} \subset \mathbb{R}^n$ ένα δικτυωτό διάστασης n και $S \subset \mathbb{R}^n$ ένα συμμετρικό κυρτό σύνολο για το οποίο

$$volS > 2^n \det(\mathcal{L}) .$$

Τότε, το σύνολο S περιέχει ένα μη μηδενικό διάνυσμα του δικτυωτού.

Το προηγούμενο θεώρημα είναι γνωστό ως *Θεώρημα του Minkowski*.

Έστω $\mathcal{L}$ ένα δικτυωτό διάστασης n και $\vec{v_s}$ ένα μικρότερο διάνυσμα αυτού. Η υπόθεση ότι για κάθε $\epsilon > 0$ ισχύει:

$$(1-\epsilon)\, Gauss(\mathcal{L}) \leq \|\vec{v_s}\| \leq (1+\epsilon)\, Gauss(\mathcal{L})$$

δηλαδή

$$\|\vec{v_s}\| \approx Gauss(\mathcal{L}),$$

ονομάζεται *ευρετική Gauss*.

Το αποτέλεσμα αυτό, δεν έχει αποδειχθεί μέχρι σήμερα, ωστόσο όπως θα δούμε στη συνέχεια, μπορεί να χρησιμοποιηθεί για ορισμένους τύπους δικτυωτών. Για περισσότερες πληροφορίες σχετικά με την μαθηματική αιτιολόγηση της ευρετικής αυτής αρχής, ο ενδιαφερόμενος αναγνώστης μπορεί να ανατρέξει στην βιβλιογραφία [43].

Θεώρημα 7.3. Έστω $\mathcal{B} : \vec{v_1}, \ldots, \vec{v_n}$ μια βάση ενός δικτυωτού $\mathcal{L}$ σε έναν $m-$διάστατο χώρο $V \subseteq \mathbb{R}^m$. Τότε, τα διανύσματα $\vec{v_1^*}, \ldots, \vec{v_n^*}$ με

$$\vec{v_1^*} = \vec{v_1} \ , \ \vec{v_i^*} = \vec{v_i} - \sum_{j=1}^{i-1} \mu_{i,j} \vec{v_j^*} \ , \ i = 2, \ldots, n$$

και

$$\mu_{i,j} = \vec{v_i} \cdot \vec{v_j^*} / \left\| \vec{v_j^*} \right\|^2 \ , \ 1 \leq j < i$$

αποτελούν μια ορθογώνια βάση του V. Η ορθογώνια αυτή βάση, μπορεί να μετατραπεί σε ορθοκανονική, διαιρώντας κάθε διάνυσμα με την Ευκλείδεια νόρμα του.

Το θεώρημα αυτό είναι γνωστό ως *Θεώρημα Gram–Schmidt*.

7.3 Δικτυωτά στην Κρυπτανάλυση

Στην ενότητα αυτή θα αναλύσουμε τις σημαντικότερες τεχνικές ιδιότητες των δικτυωτών, οι οποίες συμβάλουν στις κρυπταναλυτικές επιθέσεις που θα εφαρμόσουμε στη συνέχεια.

Στα επόμενα, ένα διάνυσμα $\vec{v}$, θα συμβολίζεται ποιο σύντομα με $\mathbf{v}$.

Λήμμα 7.1. Έστω $q \in \mathbb{P}$ με $q > 2$ και $n, A_i \in \mathbb{N}$ με $i = 1, \ldots, n$ για τα οποία ισχύει:

$$0 < n \leq \log\log q - 1 \ \ , \ \ 2^{i-1} q^{i/(n+1)} < A_i < 2^i q^{i/(n+1)}.$$

Αν $\mathcal{L}$ είναι ένα πλήρες δικτυωτό το οποίο παράγεται από τις γραμμές του πίνακα

$$\begin{pmatrix} -1 & A_1 & A_2 & \cdots & A_n \\ 0 & q & 0 & \cdots & 0 \\ 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & q \end{pmatrix},$$

τότε για κάθε μη μηδενικό διάνυσμα $\mathbf{v} \in \mathcal{L}$, ισχύει:

$$\|\mathbf{v}\| > \frac{q^{n/(n+1)}}{8}.$$

Απόδειξη. Αρχικά αποδεικνύεται ότι $A_n < q$. Έχουμε:

$$n(n+1) \leq (\log\log q - 1)(\log\log q) \leq (\sqrt{\log q} - 1)\sqrt{\log q} < \log_2 q$$

επομένως

$$n(n+1) < \log_2 q \Leftrightarrow 2^n < q^{1/(n+1)} \Leftrightarrow 2^n q^{n/(n+1)} < q$$

και συνεπώς $A_n < 2^n q^{n/(n+1)} < q$.

Στη συνέχεια, υποθέτουμε ότι υπάρχουν ακέραιοι αριθμοί $x_0, \ldots, x_n$, όχι όλοι μηδέν, τέτοιοι ώστε:

$$\mathbf{v} = (-x_0, x_0A_1 + x_1q, \ldots, x_0A_n + x_nq) \in \mathcal{L} \text{ με } \|\mathbf{v}\| \leq \frac{q^{n/(n+1)}}{8} < q.$$

Τότε έχουμε:

$$\max\{|x_0|, |x_0A_1 + x_1q|, \ldots, |x_0A_n + x_nq|\} < \frac{q^{n/(n+1)}}{8}.$$

Αν $x_0 = 0$, τότε $\|\mathbf{v}\| = q\sqrt{x_1^2 + \cdots + x_n^2} \neq 0$ και επομένως υπάρχει τουλάχιστον ένα $i \in \{1, \ldots, n\}$ τέτοιο ώστε $x_i \neq 0$. Τότε όμως, $\|\mathbf{v}\| \geq q$, το οποίο είναι άτοπο.Έτσι $x_0 \neq 0$. Αν $x_0A_j + x_jq = 0$ για κάποιο $j \in \{1, \ldots, n\}$, τότε $x_j \neq 0$. Έτσι $q \mid x_0A_j$ και επομένως $q \mid x_0$ ή $q \mid A_j$. Όμως, $0 < |x_0| < q$ και $0 < A_j \leq A_n < q$ το οποίο είναι άτοπο.Έτσι $x_0A_j + x_jq \neq 0$ $(j = 1, \ldots, n)$.

Έστω $i \in \{1, \ldots, n\}$ τέτοιο ώστε:

$$\frac{q^{(n-i)/(n+1)}}{2^{i+2}} < |x_0| < \frac{q^{(n-i+1)/(n+1)}}{2^{i+1}}.$$

Αν $x_j \neq 0$ με $1 \leq j \leq i$, τότε έχουμε:

$$\frac{q^{n/(n+1)}}{8} > |x_0A_j + x_jq| \geq q - |x_0| A_j > q - \frac{q^{(n-i+1)/(n+1)}}{2^{i+1}} A_i > \frac{q}{2},$$

από όπου παίρνουμε $1 > 4q^{1/(n+1)}$ το οποίο είναι άτοπο. Επομένως $x_j = 0$ $(j = 1, \ldots, i)$ και συνεπώς

$$\|\mathbf{v}\| > |x_0 A_i| > \frac{q^{(n-i)/(n+1)}}{2^{i+2}} 2^{i-1} q^{i/(n+1)} = \frac{q^{n/(n+1)}}{8}$$

το οποίο επίσης είναι άτοπο. □

Θεώρημα 7.1. Έστω $\mathcal{L}$ ένα $n-$ διάστατο δικτυωτό και $\mathbf{y} \in \mathbb{R}^n$. Τότε υπάρχει ντετερμινιστικός αλγόριθμος ο οποίος υπολογίζει ένα διάνυσμα $\mathbf{v} \in \mathcal{L}$, τέτοιο ώστε για κάθε άλλο διάνυσμα $\mathbf{t} \in \mathcal{L}$, να ισχύει:

$$\|\mathbf{v} - \mathbf{y}\| \leq \|\mathbf{t} - \mathbf{y}\|$$

Η πολυπλοκότητα χρόνου αυτού του αλγόριθμου, είναι $2^{2n+o(n)}$.

Απόδειξη. Βλέπε [28]. □

Θεώρημα 7.2. Έστω $q \in \mathbb{P}$ με $q > 2$ και $n, A_i, B_i \in \mathbb{N}$ με $i = 1, \ldots, n$ για τα οποία ισχύει $0 < n \leq \log\log q - 1$ και $2^{i-1} q^{i/(n+1)} < A_i < 2^i q^{i/(n+1)}$. Τότε το σύστημα των ισοτιμιών

$$y_i + A_i x + B_i \equiv 0 \bmod q \quad (i = 1, \ldots, n) \tag{7.1}$$

έχει το πολύ μια λύση $\mathbf{v} = (x, y_1, \ldots, y_n) \in \{0, \ldots, q-1\}^{n+1}$ με

$$\|\mathbf{v}\| < \frac{q^{n/(n+1)}}{16}.$$

Εάν υπάρχει μια τέτοια λύση, τότε ο αλγόριθμος του Θεωρήματος 7.1, για $\mathbf{y} = (0, B_1, \ldots, B_n)$ και για κάθε διάνυσμα $\mathbf{t}$ του δικτυωτού $\mathcal{L}$ του Λήμματος 7.1, υπολογίζει ένα διάνυσμα $\mathbf{w}$, όπου η πρώτη του συντεταγμένη είναι ο αριθμός x. Η πολυπλοκότητα χρόνου υπολογισμού του x, είναι $O((\log q)^2)$.

Απόδειξη. Έστω $\mathbf{v} = (x, y_1, \ldots, y_n)$ μια λύση του συστήματος (7.1) με $\mathbf{v} < \frac{q^{n/(n+1)}}{16}$. Θεωρούμε $\mathcal{L}$ ένα πλήρες δικτυωτό, το οποίο παράγεται από τις γραμμές του πίνακα

$$\begin{pmatrix} -1 & A_1 & A_2 & \cdots & A_n \\ 0 & q & 0 & \cdots & 0 \\ 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & q \end{pmatrix}$$

και $b = (0, B_1, \ldots, B_n)$. Εφόσον $y_i + A_i x + B_i \equiv 0 \bmod q$ υπάρχουν $z_i \in \mathbb{Z}$ τέτοιοι ώστε $-xA_1 + z_i q = y_i + B_i$ $(i = 1, \ldots, n)$. Τότε για το διάνυσμα $\mathbf{u} = (x, -xA_1 + z_1 q, \ldots, -xA_n + z_n q) \in \mathcal{L}$ έχουμε:

$$\|\mathbf{u} - \mathbf{b}\| = \|(x, y_1, \ldots, y_n)\| = \|\mathbf{v}\| < \frac{q^{n/(n+1)}}{16}.$$

Από την άλλη μεριά υπολογίζουμε ένα διάνυσμα $\mathbf{w} \in \mathcal{L}$ τέτοιο ώστε:

$$\|\mathbf{w} - \mathbf{b}\| \leq \|\mathbf{u} - \mathbf{b}\| < \frac{q^{n/(n+1)}}{16}$$

και έτσι έχουμε:

$$\|\mathbf{u} - \mathbf{w}\| \leq \|\mathbf{u} - \mathbf{b}\| + \|\mathbf{w} - \mathbf{b}\| < \frac{q^{n/(n+1)}}{8}.$$

Εφόσον $\mathbf{u} - \mathbf{w} \in \mathcal{L}$, από το Λήμμα 7.1 παίρνουμε $\mathbf{u} = \mathbf{w}$ και έτσι $\mathbf{w} = (w_0, \ldots, w_n)$ όπου $w_0 = x$. □

Λήμμα 7.2. Έστω $h(x, y) \in \mathbb{R}[x, y]$ το σύνολο w μονωνύμων. Έστω επίσης ότι $X, Y \in \mathbb{R}^+$ και ακέραιοι x_0, y_0 τέτοια ώστε $|x_0| < X$, $|y_0| < Y$. Αν ισχύουν:

1. $h(x_0, y_0) \in \mathbb{Z}$,
2. $\|h(xX, yY)\| = \sqrt{\sum_{i,j} \left(h_{i,j} X^i Y^j\right)^2} < \frac{1}{\sqrt{w}}$,

τότε $h(x_0, y_0) = 0$

Απόδειξη. [20] □

Ορισμός 7.1. Έστω $\mathcal{B} : \mathbf{b}_1, \ldots, \mathbf{b}_w$ μια βάση ενός δικτυωτού $\mathcal{L}$ σε έναν $m-$διάστατο χώρο $V \subseteq \mathbb{R}^m$ και $\mathcal{B}^* : \mathbf{b}_1^*, \ldots, \mathbf{b}_w^*$ μια ορθογώνια βάση του V. Η βάση $\mathcal{B}$ ονομάζεται *LLL-μειωμένη βάση* του $\mathcal{L}$ εάν ικανοποιούνται οι δύο επόμενες προϋποθέσεις:

Συνθήκη Μεγέθους:

$$|\mu_{i,j}| = \frac{|\mathbf{b}_i \mathbf{b}_j^*|}{\left\|\mathbf{b}_j^*\right\|^2} \leq \frac{1}{2} \ , \ \forall \ 1 \leq j < i \leq w$$

.

Συνθήκη Lovasz:

$$\|\mathbf{b}_i^*\|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) \left\|\mathbf{b}_{i-1}^*\right\|^2 \ , \ \forall\ 1 < i \leq w$$

.

Λήμμα 7.3. Έστω $\mathcal{L}$ ένα δικτυωτό και $\mathcal{B} : \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_w$ μια LLL-μειωμένη βάση του. Τότε,

$$\|\mathbf{b}_1\| < 2^{(w-1)/4}(\det \mathcal{L})^{1/w} \ , \ \|\mathbf{b}_2\| \leq 2^{w/4} \left(\frac{\det \mathcal{L}}{\|\mathbf{b}_1\|}\right)^{1/(w-1)} \ , \ \|\mathbf{b}_2\| < \frac{3}{2} \|\mathbf{b}_1\|$$

Απόδειξη. Για τις πρώτες δύο ανισότητες βλέπε [20]. Για την τρίτη ανισότητα έχουμε $\|\mathbf{b}_2\| < \|\mathbf{b}_2^*\| + \frac{1}{2}\|\mathbf{b}_1\|$ και $\|\mathbf{b}_2^*\| < \|\mathbf{b}_1\|$ από τα οποία προκύπτει το ζητούμενο. □

Κεφάλαιο 8

Σύγχρονες Επιθέσεις με Χρήση Δικτυωτών

Στο κεφάλαιο αυτό, μελετώνται κρυπταναλυτικές επιθέσεις κατά του σχήματος DSA καθώς και στην παραλλαγή του ECDSA. Σκοπός ενός επιτιθέμενου είναι να προσδιορίσει είτε το δημόσιο κλειδί A, είτε το εφήμερο κλειδί k, που ικανοποιούν μια εξίσωση ισοτιμίας της μορφής:

$$f_s(A, k) \equiv 0 \bmod q. \tag{8.1}$$

Τα σχήματα αυτά, στηρίζονται στην δυσκολία επίλυσης του διακριτού λογάριθμου (DLP) όπου ο επιτιθέμενος καλείται να δώσει λύση, ενώ μια διαφορετική κατηγορία από επιθέσεις, βασίζεται στην θεωρία των δικτυωτών, την οποία θα αναλύσουμε στις επόμενες δύο ενότητες.

8.1 Πρώτη Επίθεση στην Υπογραφή (EC)DSA

8.1.1 Σχεδίαση Αλγόριθμου Επίθεσης

Έστω n ένας θετικός ακέραιος με $0 < n \leq \log\log q - 1$. Υποθέτουμε, ότι έχουμε στη διάθεσή μας t ακριβώς υπογεγραμμένα μηνύματα M_j με την υπογραφή DSA (αντίστοιχα ECDSA), όπου $t \leq n$ και $j = 1, \ldots, t$ με τις αντίστοιχες υπογραφές τους (r_j, s_j). Τότε υπάρχουν ακέραιοι $k_j \in \{1, \ldots, q-1\}$ τέτοιοι ώστε $r_j = (g^{k_j} \bmod p) \bmod q$ (αντίστοιχα $r_j = x_j \bmod q$ όπου $(x_j, y_j) = k_j P$) και $s_j = k_j^{-1}(\mathcal{H}(M_j) + A r_j) \bmod q$. Επομένως έχουμε:

$$k_j + C_j A + D_j \equiv 0 \bmod q \ , \ j = 1, \ldots, t \tag{8.2}$$

όπου $C_j = -r_j s_j^{-1} \bmod q$ και $D_j = -s_j^{-1}\mathcal{H}(M_j) \bmod q$. Οι προηγούμενες ισοτιμίες, σχηματίζουν ένα σύστημα με αγνώστους τους ακεραίους k_j

όπου $(j = 1, \ldots, t)$ και το μυστικό κλειδί A.

Στη συνέχεια, το σύστημα (8.2), θα τροποποιηθεί κατάλληλα, σε 4 ισοδύναμα συστήματα, βάση των οποίων θα περιγράψουμε τον αλγόριθμο επίθεσης κατά της υπογραφής DSA και ECDSA. Αρχικά, επιλέγουμε ακέραιους A_i $(i = 1, \ldots, n)$ με $2^{i-1}q^{i/(n+1)} < A_i < 2^i q^{i/(n+1)}$.

1. Πολλαπλασιάζοντας το σύστημα (8.2) με $A_i C_j^{-1}$ παίρνουμε το ισοδύναμο σύστημα

$$\underbrace{k_j A_i C_j^{-1}}_{y_{i,j}} + A_i \underbrace{A}_{x} + \underbrace{D_j A_i C_j^{-1}}_{B_{i,j}} \equiv 0 \bmod q \tag{8.3}$$

   με $i = 1, \ldots, n$ και $j = 1, \ldots, t$.

2. Πολλαπλασιάζοντας το σύστημα (8.2) με $A^{-1} A_i D_j^{-1}$ παίρνουμε το ισοδύναμο σύστημα

$$\underbrace{k_j A^{-1} A_i D_j^{-1}}_{y_{i,j}} + A_i \underbrace{A^{-1}}_{x} + \underbrace{C_j A_i D_j^{-1}}_{B_{i,j}} \equiv 0 \bmod q \tag{8.4}$$

   με $i = 1, \ldots, n$ και $j = 1, \ldots, t$.

3. Από το σύστημα (8.2) για $j = t$ παίρνουμε

$$A \equiv -C_t^{-1}(k_t + D_t) \bmod q.$$

   Πολλαπλασιάζοντας την ισοτιμία αυτή με C_j και ακολούθως προσθέτοντας $k_j + D_j$, παίρνουμε το σύστημα

$$k_j + C_j A + D_j \equiv k_j - C_j C_t^{-1}(k_t + D_t) + D_j \bmod q$$

   με $j = 1, \ldots, t-1$. Από το σύστημα αυτό και το (8.2) και σύμφωνα με την Ιδιότητα 1 της Πρότασης 2.3, προκύπτει το σύστημα

$$k_j - C_j C_t^{-1}(k_t + D_t) + D_j \equiv 0 \bmod q \tag{8.5}$$

   με $j = 1, \ldots, t-1$. Πολλαπλασιάζοντας το τελευταίο αυτό σύστημα με $-A_i C_j^{-1} C_t$ καταλήγουμε στο σύστημα

$$\underbrace{-k_j A_i C_j^{-1} C_t}_{y_{ij}} + A_i \underbrace{k_t}_{x} + \underbrace{A_i(D_t - D_j C_j^{-1} C_t)}_{B_{ij}} \equiv 0 \bmod q \tag{8.6}$$

   με $i = 1, \ldots, n$ και $j = 1, \ldots, t-1$.

4. Πολλαπλασιάζοντας το σύστημα (8.5) με $k_t^{-1}A_iE_j$ παίρνουμε το ισοδύναμο σύστημα

$$\underbrace{k_jk_t^{-1}A_iE_j}_{y_{i,j}} + A_i \underbrace{k_t^{-1}}_{x} + \underbrace{(-C_jC_t^{-1}A_iE_t)}_{B_{i,j}} \equiv 0 \bmod q \qquad (8.7)$$

με $i = 1, \ldots, n$ και $j = 1, \ldots, t-1$ όπου $E_j = -C_j^{-1}C_tD_t^{-1} + D_j^{-1}$.

8.1.2 Αλγόριθμος Πρώτης Επίθεσης

Στη συνέχεια, παρουσιάζουμε τον αλγόριθμο επίθεσης, ο οποίος βασίζεται στο Θεώρημα 7.2 και υπολογίζει την τιμή του ιδιωτικού κλειδιού A, κάτω από ορισμένες προϋποθέσεις.

Αλγόριθμος 8.1. ECDSA-ATTACK-1

Είσοδος: Οι ακέραιοι M_j, r_j, s_j με $j = 1, \ldots, t$.
Έξοδος: Το ιδιωτικό κλειδί A.

1. Υπολογίζονται $C_j = -r_js_j^{-1} \bmod q$ και $D_j = -s_j^{-1}\mathcal{H}(M_j) \bmod q$ για κάθε $j = 1, \ldots, t$.

2. Επιλέγονται ακέραιοι A_i $(i = 1, \ldots, n)$ με $2^{i-1}q^{i/(n+1)} < A_i < 2^iq^{i/(n+1)}$. (Αν $2^{i-1}q^{i/(n+1)} < C_i < 2^iq^{i/(n+1)}$, τότε $A_i = C_i$).

3. Επιλέγεται ένα από τα 4 συστήματα (8.3), (8.4), (8.6) ή (8.7).

4. Υπολογίζονται οι ακέραιοι $B_{i,j}$ για κάθε $i = 1, \ldots, n$ και $j = 1, \ldots, m$ όπου $m = t$ για τα συστήματα (8.3), (8.4) και $m = t-1$ για τα συστήματα (8.6), (8.7).

5. Έστω M το σύνολο των απεικονίσεων $\mu : \{1, \ldots, n\} \to \{1, \ldots, m\}$. Για κάθε $\mu \in M$ και σύμφωνα με το Θεώρημα 7.2, υπολογίζεται ο συντελεστής x_μ της λύσης $\mathbf{v}_\mu = \left(x_\mu, y_{1,\mu(1)}, \ldots, y_{n,\mu(n)}\right)$ του συστήματος

$$y_{i,\mu(i)} + A_ix + B_{i,\mu(i)} \equiv 0 \bmod q \quad , \quad (i = 1, \ldots, n) \qquad (8.8)$$

με $\|\mathbf{v}_\mu\| < \frac{q^{n/(n+1)}}{16}$.

6. Για όλα τα x_μ που υπολογίστηκαν, ελέγχεται αν $x_\mu = A$.

Η πολυπλοκότητα χρόνου του αλγόριθμου αυτού, αναλύεται στην επόμενη πρόταση:

Πρόταση 8.1. Ο αλγόριθμος 8.1 υπολογίζει τον ακέραιο A σε χρόνο $O((\log p)^2(\log q)^{1+\log t})$ ψηφιακές πράξεις στην περίπτωση της υπογραφής DSA και σε χρόνο $O((\log q)^2((\log q)^{\log t}+(\log\log)^4))$ ψηφιακές πράξεις και $O((\log q)^{1+\log t})$ πράξεις μέσα στην ομάδα της ελλειπτικής καμπύλης, στην περίπτωση της υπογραφής ECDSA.

Απόδειξη. Η πολυπλοκότητα χρόνου του αλγόριθμου (8.1), προκύπτει από το άθροισμα του χρόνου που απαιτείται για να κατασκευαστούν τα συστήματα ισοτιμιών (8.8), του χρόνου που απαιτείται για να βρεθούν οι λύσεις αυτών των συστημάτων και του χρόνου που απαιτείται για να ελεγχθούν όλες αυτές οι λύσεις.

Βήμα 1. Στο βήμα αυτό απαιτούνται $O\left((\log q)^2 t\right)$ ψ.π.

Βήμα 2. Ο υπολογισμός κάθε q^i απαιτεί $O\left(i^2(\log q)^2\right)$ ψ.π. Ο υπολογισμός του $\left\lfloor q^{i/(n+1)}\right\rfloor$ απαιτεί $O\left(i^2(\log q)^2\right)$ ψ.π και του $2^i\left\lfloor q^{i/(n+1)}\right\rfloor$ $O\left(i^3(\log q)^2\right)$ ψ.π. Έτσι μπορούμε να επιλέξουμε τους ακεραίους A_i ώστε

$$2^{i-1}\left(\left\lfloor q^{i/(n+1)}\right\rfloor+1\right)\leq A_i\leq 2^i\left\lfloor q^{i/(n+1)}\right\rfloor$$

και επομένως η πολυπλοκότητα χρόνου του βήματος αυτού είναι

$$O\left(\sum_{i=1}^{n} i^3(\log q)^2\right)=O\left((\log q)^2 n^4\right)=O\left((\log q)^2(\log\log q)^4\right).$$

Βήμα 4. Στο βήμα αυτό, απαιτούνται $O\left((\log q)^2(\log\log q)^2\right)$ ψ.π.

Βήμα 5. Στο βήμα αυτό, εφαρμόζεται ο αλγόριθμος του Θεωρήματος 7.1 ακριβώς t^n φορές και έτσι απαιτούνται $O\left((\log q)^{2+\log t}\right)$ ψ.π. Επομένως, η πολυπλοκότητα χρόνου της κατασκευής όλων των συστημάτων και του υπολογισμού του x_μ, είναι:

$$O\left((\log q)^2\left((\log q)^{\log t}+(\log\log q)^4\right)\right).$$

Βήμα 6. Η πολυπλοκότητα χρόνου εδώ είναι $O\left((\log p)^2(\log q)^{1+t}\right)$ ψ.π στην περίπτωση της υπογραφής DSA και $O((\log q)^{1+t})$ π.ε.κ στην περίπτωση της υπογραφής ECDSA.

Συνεπώς, στην περίπτωση της υπογραφής DSA απαιτούνται

$$O((\log p)^2(\log q)^{1+\log t}) \text{ ψ.π}$$

ενώ στην περίπτωση της υπογραφής ECDSA απαιτούνται

$$O((\log q)^2((\log q)^{\log t} + (\log\log q)^4)) \text{ ψ.π και } O((\log q)^{1+\log t}) \text{ π.ε.κ.}$$

□

8.1.3 Παράδειγμα Πρώτης Επίθεσης

Χρησιμοποιώντας τις εντολές του προηγούμενου κεφαλαίου με τον τρόπο που εφαρμόστηκαν, επιλέγουμε τον πρώτο $p = 2^{160}+7$ και την ελλειπτική καμπύλη mod p:

$$E : y^2 \equiv x^3 + 10x + 1343632762150092499701637438970764818528075565078$$

η οποία έχει τάξη

$$|E(\mathbb{Z}_p)| = 1461501637330902918203683518218126812711137002561.$$

Στη συνέχεια επιλέγουμε $q = |E(\mathbb{Z}_p)|$ και το ιδιωτικό κλειδί

$$A = 874984668032211733311386841306673749333236586178.$$

Το δημόσιο κλειδί P, Q είναι:

$$\begin{aligned} X_P &= 858713481053070278779168032920613680360047535271 \\ Y_P &= 364938321350392265038182051503279726748224184066 \\ X_Q &= 597162246892872056034315330452950636324741691536 \\ Y_Q &= 118187732920835306056696926675892475754968435739 0 \end{aligned}$$

Επειδή

$$\log\log q - 1 \approx 4.7$$

θα χρησιμοποιήσουμε τρία μηνύματα για την επίθεση αυτή, επομένως $n = t = 3$. Έστω $M_i, i = 1, \ldots, 3$ τρία μηνύματα για τα οποία ισχύουν:

$$\begin{aligned} \mathcal{H}(M_1) &= 90512292495625974813711378494310668855393091632 6 \\ \mathcal{H}(M_2) &= 14063329655378920188850486603590268110975209165 19 \\ \mathcal{H}(M_3) &= 91049725102251114155851279607869642532615044536 4 \end{aligned}$$

Επιλέγουμε τα τρία επόμενα εφήμερα κλειδιά

$$\begin{aligned} k_1 &= 46608054332288968883546711583551839882652375003 1 \\ k_2 &= 73075081866545145910184241635814150982796627158 9 \\ k_3 &= 73075081866545145910184241635814150982796627968 1 \end{aligned}$$

με τα οποία λαμβάνουμε τις υπογραφές

$$\begin{aligned} r_1 &= 12541577290894439954181238325238082770313139494 62 \\ r_2 &= 72514437791024688553461670675669940419550766323 1 \\ r_3 &= 25059359814785811483691313826556491545746471085 1 \\ s_1 &= 18878865857587237319953489809105835364056821212 3 \\ s_2 &= 10318896085393272335462652765259068309612300505 03 \\ s_3 &= 17166507828745705549481739178193859485428063759 8 \end{aligned}$$

Οι τιμές των C_j, D_j,$j = 1, \ldots, 3$ είναι:

$$\begin{aligned} C_1 &= 13804849616112617699476615342182388406507951953 34 \\ C_2 &= 38737483416567371305328231959974684116828497875 8 \\ C_3 &= 22194017158494640329963270461855126300402409283 3 \\ D_1 &= 33846469201820201955364805435268472186715989139 0 \\ D_2 &= 11799076530517507229264783205627063940991831649 22 \\ D_3 &= 40249098529983317813932909531308508838686702525 6 \end{aligned}$$

Αν τώρα επιλέξουμε τα συστήματα της μορφής (8.4) και

$$\begin{aligned} A_1 &= 1653286617349 \\ A_2 &= 27692817367581338173049 17 \\ A_3 &= 75848536991213595772910759498450120 85 \end{aligned}$$

τότε το σύστημα

$$\begin{aligned} y_{1,1} + A_1 x + B_{1,1} &\equiv 0 \bmod p \\ y_{2,2} + A_2 x + B_{2,2} &\equiv 0 \bmod p \\ y_{3,3} + A_3 x + B_{3,3} &\equiv 0 \bmod p \end{aligned}$$

δίνει λύση

$$\begin{aligned} x &= 5070602400912917605986812821509 \\ y_{1,1} &= 1317 \\ y_{2,2} &= 159 \\ y_{3,3} &= 43 \end{aligned}$$

με

$$\begin{aligned} B_{1,1} &= 14614932541718115911681851979331660384740172416 03 \\ B_{2,2} &= 13435638359300703398049990713736068019758057130 85 \\ B_{3,3} &= 13995610908559214571260961142584812708407617310 45 \end{aligned}$$

και

$$\sqrt{x^2 + y_{1,1}^2 + y_{2,2}^2 + y_{3,3}^2} < \frac{q^{3/4}}{16}.$$

Επομένως, σύμφωνα με το Θεώρημα 7.2 αποκαλύπτεται η τιμή $A^{-1} \bmod q$ και κατά συνέπεια το ιδιωτικό κλειδί A.

8.2 Δεύτερη Επίθεση στην Υπογραφή ECDSA

Στην ενότητα αυτή, περιγράφεται μια επίθεση στην ψηφιακή υπογραφή ECDSA η οποία βασίζεται στην μέθοδο του Coppersmith. Συγκεκριμένα αποδεικνύεται ότι, αν A, k είναι το ιδιωτικό και το εφήμερο κλειδί αντίστοιχα, του σχήματος ψηφιακής υπογραφής ECDSA και ισχύει:

$$(k^{-1} \bmod q)^2 A < 0.262 \cdot q^{1.157}$$

τότε είναι δυνατόν, να βρεθεί με αποτελεσματικό τρόπο, η τιμή του A.

Η παρούσα επίθεση, βασίζεται στην θεωρία των δικτυωτών μελετώντας την εξίσωση ισοτιμίας

$$ks - Ar \equiv \mathcal{H}(M) \bmod q \tag{8.9}$$

όπου $\mathcal{H} : G \to \mathbb{Z}_q$ είναι μια δημόσια γνωστή συνάρτηση κατακερματισμού, $r = (g^k \bmod p) \bmod q$ και s ικανοποιεί την εξίσωση (8.9). Το ζεύγος (r, s) είναι η υπογραφή του μηνύματος M.

8.2.1 Σχεδίαση Επίθεσης

Η πρώτη επίθεση στην ψηφιακή υπογραφή ECDSA η οποία χρησιμοποιεί την μέθοδο του Coppersmith, υπάρχει στην βιβλιογραφία [5]. Οι συγγραφείς αυτής της επίθεσης, κατάφεραν να αποδείξουν, ότι αν ισχύει: $Ak < q^{0.957}$ (με q των 160-bits), τότε υπάρχει ένας αποδοτικός αλγόριθμος που υπολογίζει το A. Στην πράξη, εφάρμοσαν την μέθοδο του Coppersmith, στο πολυώνυμο που δίνεται από την εξίσωση (8.9). Η μέθοδος του Coppersmith, έχει πολυωνυμική πολυπλοκότητα χρόνου, διότι χρησιμοποιεί τον αλγόριθμο LLL.

Παρατήρηση 8.1. Πριν συνεχίσουμε την περιγραφή της επίθεσης, θα επισημάνουμε ορισμένες παρατηρήσεις.

- Αν n είναι ένας ακέραιος με $\mu\kappa\delta\,(q, n) = 1$, τότε συμβολίζουμε πιο σύντομα με $[n^{-1}]_q$ τον ακέραιο $n^{-1} \bmod q$.

- Η μέθοδος του Coppersmith εφαρμόζεται σε ένα τετραγωνικό πολυώνυμο.

- Υποθέτοντας ότι είναι δυνατή η παραγοντοποίηση ενός ακεραίου μεγέθους μέχρι 160-bits και ότι ισχύει $[k^{-1}]_q^2 A < q/6^{3/2} \approx 0.06 \cdot q$, οι συγγραφείς της επίθεσης [5] βρίσκουν την τιμή του A σε πολυωνυμικό χρόνο.

- Το προηγούμενο αποτέλεσμα, είναι σημαντικά βελτιωμένο σε αυτήν την επίθεση που θα περιγράψουμε.

- Εάν το A είναι μικρότερο από 101-bits τότε μπορούν να χρησιμοποιηθούν τιμές μεγαλύτερες από $[k^{-1}]_q$. Για τον σκοπό αυτό, χρησιμοποιούμε τα δικτυωτά τύπου Boneh-Durfee type [20].

- Η παρούσα επίθεση, δεν στηρίζεται στην υπόθεση της παραγοντοποίησης ακεραίων με μέγεθος 160-bits και έτσι το μέγεθος του q μπορεί να είναι περισσότερο από 160-bits.

Για την επίθεση, χρησιμοποιούμε ένα δικτυωτό, τέτοιο ώστε κάθε γραμμή του να αντιστοιχεί σε ένα πολυώνυμο δύο μεταβλητών, $H(x, y)$ με $H([k^{-1}]_q, A) \in \mathbb{Z}$. Έχοντας δύο μικρά διανύσματα του δικτυωτού, παίρνουμε δύο πολυώνυμα τα οποία έχουν κοινή ρίζα την $([k^{-1}]_q, A)$. Στη συνέχεια, υπολογίζεται το ιδιωτικό κλειδί A. Για να εφαρμοστεί η μέθοδος αυτή, χρησιμοποιείται η ακόλουθη υπόθεση:

Έστω ότι έχουμε δύο αλγεβρικά ανεξάρτητα πολυώνυμα $H_1(x, y)$ και $H_2(x, y)$. Λαμβάνοντας την απαλοίφουσα των δύο αυτών πολυωνύμων (είτε ως προς x είτε ως προς y), παίρνουμε ένα μη σταθερό πολυώνυμο με μια μεταβλητή.

Επιπλέον, χρησιμοποιούμε το ακόλουθο πειραματικό γεγονός:
Γεγονός 1: *Σε τυχαία δικτυωτά διάστασης* ≤ 35 *ο LLL συμπεριφέρεται ως SVP.*

Το γεγονός αυτό, οφείλεται στην υπόθεση, ότι δεν υπάρχουν προσεγγιστικοί πολυωνυμικοί αλγόριθμοι που να δέχονται ως είσοδο ένα δικτυωτό και να εξάγουν ένα από τα μικρότερα διανύσματά του, όταν η διάσταση του δικτυωτού είναι πολύ μεγάλη. Ωστόσο, για μικρές τιμές της διάστασης του δικτυωτού ($n < 40$) ο LLL εγγυάται την εύρεση ενός μικρού διανύσματος του δικτυωτού σε πολυωνυμικό χρόνο, πράγμα το οποίο έχει αποδειχθεί σε πολλές δοκιμές [32]. Στη συνέχεια αποδεικνύεται η ακόλουθη πρόταση:

Πρόταση 8.2. Έστω A, k το ιδιωτικό και το εφήμερο κλειδί, αντίστοιχα, του σχήματος ψηφιακής υπογραφής ECDSA και $X, Y \in \mathbb{Z}$ τέτοια ώστε $[k^{-1}]_q < X, A < Y$. Λαμβάνοντας υπόψη το παραπάνω Γεγονός 1, αν ισχύει $X^2 Y^{1.26} < 0.262 \cdot q^{1.157}$ τότε είναι δυνατόν να προσδιορισθεί αποτελεσματικά το ιδιωτικό κλειδί A.

Το αποτέλεσμα της επίθεσης [36], για τιμές του Y μεγέθους μικρότερο από 101 bits, έχει βελτιωθεί και φαίνεται από τα δεδομένα του επόμενου πίνακα 1. Τέλος, παρατηρούμε ότι για την απόδειξη της Πρότασης 8.2 υποθέσαμε ότι η ευρετική Gauss, στηρίζεται στα δικτυωτά που αναφέρονται στο άρθρο [3]. Η ευρετική Gauss προβλέπει το ακόλουθο φράγμα για το πρώτο διαδοχικό ελάχιστο $\lambda_1(\mathcal{L}) \approx \sqrt{\frac{w}{2\pi e}} \det \mathcal{L}^{1/w} = Gauss(\mathcal{L})$, όπου $\mathcal{L}$ είναι ένα πλήρες δικτυωτό με διάσταση w. Αυτή η ευρετική αρχή, ελέγχθηκε τρέχοντας 1000 τυχαία δείγματα δικτυωτών και έτσι προέκυψε η σχέση

$$|\lambda_1(\mathcal{L}) - Gauss(\mathcal{L})| < 10^{-2}.$$

bits(Y)	Αποτέλεσμα (σε bits)
100	1
93	3
89	4
85	5
77	7
73	8
69	9
66	10

Πίνακας 1: Στην δεύτερη στήλη υπολογίζεται η ποσότητα

$$\left\lfloor \log_2(q^{1.157} Y^{-1.26} 0.262) \right\rfloor - \left\lfloor \log_2(q Y^{-1} 6^{-3/2}) \right\rfloor$$

όπου q τον 160-bits. Έτσι από αυτή την μέθοδο, λαμβάνουμε την διαφορά (σε bits) για το X^2 σε σύγκριση με την μέθοδο του άρθρου [36].

Θεώρημα 8.1. Έστω A, k το ιδιωτικό και το εφήμερο κλειδί, αντίστοιχα, του σχήματος ψηφιακής υπογραφής ECDSA και m, t θετικοί ακέραιοι αντίστοιχα. Έστω επίσης $X, Y \in \mathbb{Z}$ έτσι ώστε $[k^{-1}]_q < X, A < Y$. Αν

$$X^2 Y^{1+\gamma(t,m)} < \left(\zeta(w) q^{\alpha(m)+\beta(m)t}\right)^{1/(\alpha(m)+\beta(m)t)} \tag{8.10}$$

όπου

$$\alpha(m) = \frac{m\,(m+1)\,(m+2)}{6}\ ,\ \beta(m) = \frac{m\,(m+1)}{2}$$

$$w = \frac{(m+1)(m+2)}{2} + t(m+1) \ , \ \gamma(t,m) = \frac{\beta(m)t^{\frac{m+t+1}{m}} - \frac{1}{2}}{\alpha(m) + \beta(m)t/2}$$

και

$$\zeta(w) = 2^{-w^2/4} w^{-w/2}$$

τότε για αρκετά μεγάλο m, είναι δυνατόν να βρεθούν, δύο πολυώνυμα $H_1(x,y)$ και $H_2(x,y)$ τέτοια ώστε $H_1([k^{-1}]_q, A) = H_2([k^{-1}]_q, A) = 0$.

Για την απόδειξη αυτού του θεωρήματος, κατασκευάζεται ένα κατάλληλο δικτυωτό και στη συνέχεια εφαρμόζεται η μέθοδος του Coppersmith. Στη συνέχεια, βρίσκουμε κατάλληλες παραμέτρους m, t έτσι ώστε να επιτευχθεί ένα καλύτερο άνω φράγμα για το γινόμενο XY. Τελικά, χρησιμοποιώντας το Γεγονός 1, την ευρετική αρχή και των συνδυασμό $m = 6$ και $t = 1$ στο Θεώρημα 8.1, προκύπτει το ζητούμενο.

8.2.2 Ανάλυση Δεύτερης Επίθεσης

Ο σκοπός αυτής της ενότητας είναι να παρουσιάσει μερικά βασικά και αναγκαία αποτελέσματα, για την απόδειξη του Θεωρήματος 8.1.

Ορισμός 8.1. Έστω $f(x,y) \in \mathbb{R}[x,y]$. Ορίζουμε ως $x-$μετατόπιση του πολυωνύμου $f(x,y)$ το πολυώνυμο $g_{i,k}(x,y) = x^i f(x,y)^k$ και ως $y-$μετατόπιση του πολυωνύμου $f(x,y)$ το πολυώνυμο $h_{j,k}(x,y) = y^j f(x,y)^k$.

Για τα επόμενα δύο Λήμματα έχουμε $B = [\mathcal{H}(M) r^{-1}]_q$ και $C = [-sr^{-1}]_q$.

Λήμμα 8.1. Έστω $f(x,y) = \frac{x(y+B)+C}{q}$ και τα διανύσματα $b_{i,k} = (g_{i,k} X^i Y^k)_{i,k}$, $k = 0, 1, \ldots, m$, $i = 0, 1, \ldots, m-k$ για κάποια $m \in \mathbb{Z}^+$ και $g_{i,k}$ οι συντελεστές της $x-$ μετατόπισης του πολυωνύμου $f(x,y)$. Έστω επίσης ο ορθογώνιος πίνακας $M_m = (b_{i,k})_{i,k}$. Τότε

$$\det M_m = q^{-\alpha(m)} X^{2\alpha(m)} Y^{\alpha(m)} \ , \ \text{όπου } \alpha(m) = \frac{m(m+1)(m+2)}{6}$$

Απόδειξη. Ο πίνακας M_m είναι:

$$
\begin{array}{c}
\\
1\\
x\\
f\\
x^2\\
xf\\
f^2\\
\vdots\\
f^m
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
1 & x & xy & x^2 & x^2y & x^2y^2 & & x^my^m
\end{array}\\
\begin{pmatrix}
1 & & & & & & & \\
 & X & & & & & & \\
* & * & q^{-1}XY & & & & & \\
 & & & X^2 & & & & \\
* & * & * & * & q^{-1}X^2Y & & & \\
* & * & * & * & * & q^{-2}X^2Y^2 & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
* & * & * & * & & & & q^{-m}X^mY^m
\end{pmatrix}
\end{array}
$$

Η διάσταση του πίνακα M_m ειναι $\frac{(m+1)(m+2)}{2}$ καθώς

$$|\{(i,k) : k = 0, 1, \ldots, m, j = 0, 1, \ldots, m-k\}| = 1 + 2 + \cdots + (m+1).$$

Επειδή όμως ο M_m είναι ένας κάτω τριγωνικός πίνακας, ισχύει:

$$\det M_m = \prod_{k=0}^{m} \prod_{i=0}^{m-k} X^{k+i} Y^k q^{-k}$$

και έτσι προκύπτει το ζητούμενο. □

Λήμμα 8.2. Έστω $f(x,y) = \frac{x(y+B)+C}{q}$ και $c_{j,k} = (h_{j,k}X^jY^k)_{i,k}$, $k = 0, 1, \ldots, m$, $j = 1, 2, \ldots, t$, για κάποια $m, t \in \mathbb{Z}^+$ όπου $h_{j,k}$, οι συντελεστές της $y-$μετατόπισης του πολυωνύμου $f(x,y)$. Έστω επίσης $R_{t,m} = (c_{j,k})_{j,k}$ και $\hat{R}_{t,m}$ ο πίνακας διάστασης $(m+1)t \times (m+1)t$. Τότε

$$\det \hat{R}_{t,m} = q^{-t\beta(m)} X^{t\beta(m)} Y^{t(m+1)(m+t+1)/2} \text{ , όπου } \beta(m) = (m+1)m/2.$$

Απόδειξη. Έχουμε $\hat{R}_{t,m} =$

$$
\begin{array}{c}
\\
y\\
xf\\
\cdots\\
yf^m\\
\vdots\\
y^tf^m
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc}
y & xy^2 & x^2y^3 & x^3y^4 & & x^my^{m+1} & & x^my^{m+t}
\end{array}\\
\begin{pmatrix}
Y & & & & & & & \\
* & q^{-1}XY^2 & & & & & & \\
 & & & & & & & \\
* & * & * & * & & q^{-m}X^mY^{m+1} & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
* & * & * & * & & & & q^{-m}X^mY^{m+t}
\end{pmatrix}
\end{array}
$$

Επομένως

$$\hat{R}_{t,m} = \prod_{k=0}^{m} \prod_{j=1}^{t} q^{-k} X^k Y^{k+j}$$

όπου μετά από μερικούς υπολογισμούς, προκύπτει το ζητούμενο. □

Πόρισμα 8.1. Έστω ο πίνακας $A_{t,m} = \begin{bmatrix} M_m \\ R_{t,m} \end{bmatrix}$ όπου $M_m, R_{t,m}$ οι πίνακες οι οποίοι ορίστηκαν στα δύο προηγούμενα Λήμματα 8.1 και 8.2. Τότε

$$\det A_{t,m} = q^{-(\alpha(m)+\beta(m)t)} X^{(2\alpha(m)+\beta(m)t)} Y^{(\alpha(m)+t(m+1)(m+t+1)/2}.$$

Απόδειξη. Έχουμε $\det A_{t,m} = \det M_m \det \hat{R}_{t,m}$, επομένως το αποτέλεσμα προκύπτει εύκολα από τα Λήμματα 8.1 και 8.2. □

Πολλαπλασιάζοντας και τις δύο πλευρές της εξίσωσης 8.9 με τον ακέραιο $-(kr)^{-1} \mod q$, παίρνουμε

$$k^{-1}(A + \mathcal{H}(M)r^{-1}) + (-sr^{-1}) \equiv 0 \bmod q.$$

Εάν θέσουμε

$$f(x,y) = \frac{x(y+B)+C}{q} \tag{8.11}$$

όπου $B = [\mathcal{H}(M)r^{-1}]_q$ και $C = [-sr^{-1}]_q$, τότε θα έχουμε $f([k^{-1}]_q, A) \in \mathbb{Z}$. Έστω $\mathcal{L}$ το δικτυωτό το οποίο παράγεται από τις γραμμές του πίνακα $A_{t,m}$ του Πορίσματος 8.1 και $f(x,y)$ το πολυώνυμο που ορίζεται από την εξίσωση (8.11). Εφαρμόζουμε τον αλγόριθμο LLL στο δικτυωτό $\mathcal{L}$ και έστω b_1 το πρώτο ελάχιστο διάνυσμα. Έστω τώρα $H_1(x,y)$ το πολυώνυμο που αντιστοιχεί στο διάνυσμα b_1, το οποίο είναι:

$$H_{x,y} = b_1 \cdot \left(1, x/X, xy/XY, \ldots, x^m y^{m+t}/X^m Y^{m+t}\right).$$

Παρατηρούμε ότι $H_1([k^{-1}]_q, A) \in \mathbb{Z}$. Πράγματι, αρχικά ισχύει:

$$g_{i,k}\left([k^{-1}]_q, A\right), h_{i,k}\left([k^{-1}]_q, A\right) \text{ και } f\left([k^{-1}]_q, A\right) \in \mathbb{Z}$$

και επειδή το H_1 είναι γραμμικός συνδυασμός πολυωνύμων από $x-$μετατόπιση και $y-$μετατόπιση, προκύπτει το ζητούμενο. Επιπλέον ισχύει:

$$\|H_1(xX, yY)\| = \|b_1\| \text{ και } \|H_1(xX, yY)\| < 2^{(w-1)/4} \det(\mathcal{L})^{1/w}.$$

Αν τώρα αποδειχθεί ότι $\|H_1(xX, yY)\| \leq 1/\sqrt{w}$, τότε από το Λήμμα 7.2 θα έχουμε $H_1([k^{-1}]_q, A) = 0$. Έτσι για να ικανοποιείται η ανίσωση

$$\left\|H_1\left([k^{-1}]_q, A\right)\right\| \leq 1/\sqrt{w}$$

θα πρέπει να ισχύει η ανίσωση

$$2^{(w-1)/4} \det(\mathcal{L})^{1/w} < \frac{1}{\sqrt{w}}.$$

Αναδιατάσσοντας τις προηγούμενες ανισώσεις, έχουμε

$$\det \mathcal{L} \leq \frac{1}{2^{w(w-1)/4}\sqrt{w}^w} = \zeta_1(w).$$

Στη συνέχεια, θα χρειαστούμε ένα ελαφρώς μικρότερο φράγμα για την ορίζουσα του δικτυωτού $\mathcal{L}$ έτσι ώστε να δημιουργήσουμε ένα κατάλληλο φράγμα για το δεύτερο πολυώνυμο. Επομένως θέτουμε $\zeta(w) = \frac{1}{2^{w^2/4}\sqrt{w}^w}$ (όπου $\zeta(w) < \zeta_1(w)$). Για να ισχύει η ανίσωση $\|H_1(xX, yY)\| \leq 1/\sqrt{w}$, θα πρέπει να έχουμε

$$det\mathcal{L} \leq \frac{1}{2^{w^2/4}\sqrt{w}^w} = \zeta(w) \tag{8.12}$$

αλλά από το Πόρισμα 8.1 έπεται ότι

$$\det \mathcal{L} = q^{(-\alpha(m)+t\beta(m))}(X^2Y)^{(\alpha(m)+\frac{\beta(m)t}{2})}Y^{t\beta(m)(\frac{m+t+1}{m}-\frac{1}{2})} \tag{8.13}$$

όπου $\alpha(m) = \frac{m(m+1)(m+2)}{6}$ και $\beta(m) = \frac{m(m+1)}{2}$. Έτσι από τις σχέσεις (8.12) και (8.12) παίρνουμε

$$q^{(-\alpha(m)+t\beta(m))}(X^2Y)^{(\alpha(m)+\frac{\beta(m)t}{2})}Y^{t\beta(m)(\frac{m+t+1}{m}-\frac{1}{2})} < \zeta(w)$$

επομένως

$$X^2Y^{1+\gamma(m,t)} < \zeta(w)^{1/(\alpha(m)+t\frac{\beta(m)}{2})}q^{(\alpha(m)+t\beta(m))/(\alpha(m)+t\frac{\beta(m)}{2})} \tag{8.14}$$

όπου

$$\gamma(m,t) = \frac{t\beta(m)\left(\frac{m+t+1}{m} - \frac{1}{2}\right)}{\alpha(m) + \beta(m)t/2}.$$

Για να βρούμε ένα δεύτερο πολυώνυμο $H_2(x, y)$ τέτοιο ώστε $H_2([k^{-1}]_q) = 0$, εργαζόμαστε με το δεύτερο LLL μειωμένο διάνυσμα b_2. Εφόσον το δικτυωτό $\mathcal{L}$ παράγεται από τα πολυώνυμα των x και y μετατοπίσεων

του $f(x,y)$, έπεται ότι η ποσότητα $\|b_1\|$ θα είναι ακέραιο πολλαπλάσιο του q^{-m}. Έτσι $\|b_1\| > q^{-m}$ και από το Λήμμα 7.3 βρίσκουμε

$$\|H_2(xX, yY)\| \leq 2^{w/4} \left(\frac{\det \mathcal{L}}{\|b_1\|}\right)^{1/(w-1)} \leq 2^{w/4} q^{m/(w-1)} (\det \mathcal{L})^{1/(w-1)}.$$

Από την προηγούμενη ανισότητα και την σχέση (8.12) παίρνουμε

$$2^{w/4} q^{m/(w-1)} (\det \mathcal{L})^{1/(w-1)} \leq 2^{w/4} q^{m/(m-1)} \zeta(w)^{1/(w-1)} =$$

$$q^{\frac{m}{w-1}} w^{-\frac{w}{2(w-1)}} 2^{-\frac{w}{4(w-1)}}.$$

Για να ισχύει $H_2([k^{-1}]_q, A) = 0$, θα πρέπει να έχουμε

$$q^{\frac{m}{w-1}} w^{-\frac{w}{2(w-1)}} 2^{-\frac{w}{4(w-1)}} < w^{-1/2}.$$

Ως εκ τούτου

$$q^{m/(w-1)} < w^{-\frac{1}{2}+\frac{w}{2(w-1)}} 2^{\frac{w}{4(w-1)}} = w^{\frac{1}{2(w-1)}} 2^{\frac{w}{4(w-1)}}.$$

Συνεπώς, καταλήγουμε ότι $q^m < w^{1/2} 2^{w/4}$. Η τελευταία αυτή ανισότητα, ισχύει στην περίπτωση που το m λαμβάνει αρκετά μεγάλες τιμές ($w = O(m^2 + mt)$) το οποίο ολοκληρώνει την απόδειξη.

Στη συνέχεια, αποδεικνύουμε την Πρόταση 8.2

Εάν η διάσταση w του δικτυωτού $\mathcal{L}$ είναι ≤ 35, τότε ο αλγόριθμος LLL, στην πράξη θα επιστρέψει ένα μικρότερο διάνυσμα του δικτυωτού. Τότε το $\zeta(w)$ της σχέσης (8.12) μπορεί να αντικατασταθεί από μια μεγαλύτερη τιμή. Εάν $w \leq 35$, τότε αντικαθιστούμε το φράγμα του πρώτου LLL διανύσματος, με $\sqrt{\frac{w}{2\pi e}} \det(\mathcal{L})^{1/w}$.

Ένα μικρό διάνυσμα του δικτυωτού, έχει μέγεθος $\|\mathcal{L}\| \approx \sqrt{\frac{w}{2\pi e}} \det(\mathcal{L})^{1/w}$. Για να το δούμε αυτό, υπολογίζουμε τα πρώτα $\|\mathcal{L}\|$ ελάχιστα του δικτυωτού $\mathcal{L}$ εκτιμώντας την ακτίνα μιας σφαίρας με όγκο ίσο με $\det \mathcal{L}$. Αυτό δείχνει ότι $\|\mathcal{L}\| \approx \sqrt{\frac{w}{2\pi e}} \det(\mathcal{L})^{1/w}$. Υποθέτουμε επίσης ότι $\|H_1(xX, yY)\| < \frac{2}{3\sqrt{w}}$. Τότε από το Λήμμα 7.2 έχουμε $H_1([k^{-1}]_q, A) = 0$. Αλλά επειδή

$$\|H_1(xX, yY)\| < \frac{2}{3\sqrt{w}} < \sqrt{\frac{w}{2\pi e}} \det(\mathcal{L})^{1/w}$$

θα πρέπει

$$\sqrt{\frac{w}{2\pi e}} \det(\mathcal{L})^{1/w} < \frac{2}{3\sqrt{w}}.$$

Αναδιατάσσοντας την προηγούμενη ανίσωση, παίρνουμε

$$\det \mathcal{L} \leq \sqrt{\frac{2\pi e}{w^2}}^w \left(\frac{2}{3}\right)^w = \sqrt{\frac{8\pi e}{9w^2}}^w = \zeta'(w)$$

και ως εκ τούτου

$$X^2 Y^{1+\gamma(t,m)} < (\zeta'(w) q^{(\alpha(m)+\beta(m)t)})^{1/(\alpha(m)+\beta(m)t/2)}.$$

Τώρα, η τιμή $\zeta'(w) = \sqrt{\frac{8\pi e}{9w^2}}^w$ είναι αρκετά μεγαλύτερη από την τιμή $\zeta(w)$. Για $t = 1 \leq m \leq 6$ η διάσταση w του δικτυωτού $\mathcal{L}$ είναι ≤ 35. Χρησιμοποιώντας τις τιμές $t = 1$ και $m = 6$ παίρνουμε $X^2 Y^{1.26} < 0.262 \cdot q^{1.157}$.

Χρειαζόμαστε επίσης, ένα δεύτερο πολυώνυμο $H_2(x, y)$ τέτοιο ώστε, $H_2([k^{-1}]_q, A) = 0$. Από την τρίτη ανισότητα του Λήμματος 7.3 ισχύει:

$$\|H_2(xX, yY)\| < \frac{3}{2} \|H_1(xX, yY)\| < \frac{3}{2} \cdot \frac{2}{3\sqrt{w}} = \frac{1}{\sqrt{w}}.$$

Στη συνέχεια, από το Λήμμα 7.2 παίρνουμε και πάλι $H_2([k^{-1}]_q, A) = 0$ και από τις υποθέσεις αυτού του Λήμματος, μπορούμε να υπολογίσουμε το αποτέλεσμα (ως προς x) και στη συνέχεια τις ακέραιες ρίζες του. Μια από αυτές τις ρίζες είναι και το ιδιωτικό κλειδί A.

8.2.3 Παράδειγμα Δεύτερης Επίθεσης

Παράδειγμα 8.1. Με την εντολή numbers(1,160,160,0) παίρνουμε τον πρώτο

$$q = 1420781990420358144729370324145404355374905166249$$

τον 160-bits. Στη συνέχεια, επιλέγουμε το κλειδί

$$A = 24251561979536311495125$$

τον 75-bits και το εφήμερο κλειδί

$$k = 913551645485465300451420923974053878879771609110$$

τον 160-bits. Έστω (r, s) η υπογραφή ενός μηνύματος m και ας είναι $x(B + y) + C \equiv 0 \bmod q$ η υπογράφουσα εξίσωση με

$$\begin{aligned} B &= 269366230512225345569353296119811290445931088756 \\ C &= 542189824416770300914626882856872739739223238744 \end{aligned}$$

Θέτουμε $f(x,y) = \frac{x(y+B)+C}{q}$. Εδώ σημειώνουμε ότι $[k^{-1}]_q \cdot A > q/6^{3/2}$. Στην πραγματικότητα, έχουμε μια διαφορά μεγέθους 6-bits, διότι $[k^{-1}]_q \cdot A =$ 163-bits και $q/6^{3/2} = 157$-bits. Επομένως, σύμφωνα με τον πίνακα 1. αναμένεται να βρούμε και το ιδιωτικό κλειδί A. Εάν προσπαθήσουμε να χρησιμοποιήσουμε την μέθοδο του άρθρου [36], θα πάρουμε ένα πολυώνυμο $H(x,y)$ με $H([k^{-1}]_q, A) \neq 0$, όπως και είναι αναμενόμενο. Θεωρούμε το δικτυωτό, με παραμέτρους $t = 1$, $m = 6$ και $X = 2^{44}$, $Y = 2^{75}$ το οποίο παράγεται από τους συντελεστές των πολυωνύμων

$$\left\{1, x, xf, x^2, x^2 f, f^2, \ldots, x^6, x^6 f, \ldots, f^6, y, yf, \ldots, yf^6\right\}.$$

Εφαρμόζουμε τον αλγόριθμο LLL στις γραμμές του πίνακα διάστασης 35×35. Τα πρώτα δύο πολυώνυμα $H_1(x,y)$ και $H_2(x,y)$ που προκύπτουν είναι αντίστοιχα:

$$-5025432391212010233393120596402712690448578877 73x^6y^7 + \cdots$$

και

$$2897186270317162941899368306822192855106865698467x^6y^7 - \cdots$$

Τέλος, ένας απλός υπολογισμός της ποσότητας $R(y) = res_x(H_1, H_2)$, παρέχεται μέσο ενός μη σταθερού πολυωνύμου βαθμού 48, τέτοιο ώστε $R(A) = 0$.

Αν στο σχήμα ψηφιακής υπογραφής ECDSA εφαρμόσουμε τον αλγόριθμο Shank's baby steps-giant steps (ή τον αλγόριθμο Pollard), τότε ο διακριτός λογάριθμος του δημοσίου κλειδιού R, υπολογίζεται σε περίπου $q^{1/2}$ βήματα, όπου q είναι ο μεγαλύτερος πρώτος διαιρέτης του $p-1$. Στο παράδειγμα αυτό, το q έχει μέγεθος 160-bits, γι' αυτό είναι ανέφικτο να εφαρμοστεί αυτή η επίθεση, ακόμη και αν το μέγεθος του ιδιωτικού κλειδιού του ECDSA είναι σχετικά μικρό. Επιπλέον, επειδή στο προηγούμενο παράδειγμα το ιδιωτικό κλειδί A έχει μέγεθος 75-bits, κάθε προσπάθεια εύρεσής του με ωμή βία, θα αποτύγχανε.

8.3 Υλοποίηση Κρυπταναλυτικών Επιθέσεων

8.3.1 Συναρτήσεις Επιθέσεων

$$Attack\_Values(q, n)$$

Η συνάρτηση αυτή, δέχεται ως είσοδο τους ακέραιους αριθμούς $q > 0$ και $n \in \{2, 3\}$ και επιστρέφει για $i = 1, \ldots, n$ τις τιμές

$$\begin{aligned} L_{i_{min}} &= \left\lfloor 2^{i-1} q^{i/(n+1)} \right\rfloor \\ L_{i_{max}} &= \left\lfloor 2^{i} q^{i/(n+1)} \right\rfloor \\ l_{i_{max}} &= L_{i_{max}} - L_{i_{min}} \end{aligned}$$

Η συνάρτηση αυτή, χρησιμοποιείται προκειμένου ο χρήστης να μπορεί να επιλέγει κατάλληλα τις τιμές L_i και l_i , $i = 1, \ldots, n$ που θα χρησιμοποιήσει στην επόμενη συνάρτηση:

$$ECDSA\_ATTACK\_1$$

$$(q, a, b, p, X_p, Y_p, A, k_1, k_2, k_3, k, H_1, H_2, H_3, L_1, l_1, L_2, l_2, L_3, l_3)$$

Με την συνάρτηση αυτή, εισάγονται όλες οι τιμές των μεταβλητών q, a, b, p, X_p, Y_p, A έτσι όπως έχουν οριστεί την Ενότητα 6.4.

Οι μεταβλητές k_i, $i = 1, \ldots, 3$ είναι τα εφήμερα κλειδιά, των πολύ, τριών μηνυμάτων που θα χρησιμοποιηθούν.

Η μεταβλητή k λαμβάνει την τιμή 1, αν η επίθεση που θα εφαρμοστεί αφορά το σύστημα (8.3) ή την τιμή 2 αν η επίθεση που θα εφαρμοστεί αφορά το σύστημα (8.4).

Οι μεταβλητές H_i , $i = 1, \ldots, 3$ είναι οι τιμές κατακερματισμού των τριών μηνυμάτων. Για να εισάγει ο χρήστης τα γνωστά κείμενα που θα χρησιμοποιηθούν στην επίθεση, αρκεί να θέσει τις τιμές αυτές ίσες με 0.

Οι μεταβλητές L_i, l_i, $i = 1, \ldots, 3$ είναι θετικοί ακέραιοι και επιλέγονται κατάλληλα από τον χρήστη, έτσι ώστε, η αναζήτηση των ακεραίων A_i, $i = 1, \ldots, 3$ να πραγματοποιείται μέσα στο διάστημα

$$(L_{i_{min}} + L_i, L_{i_{min}} + L_i + l_i) \subset (L_{i_{min}}, L_{i_{max}}) .$$

Εφόσον, όλες οι μεταβλητές λάβουν τιμές, η συνάρτηση θα δώσει ως έξοδο το βέλτιστο σύστημα (8.8). Συγκεκριμένα, εξετάζονται εννέα εξισώσεις ισοτιμιών, ως εξής: Για κάθε εξίσωση ισοτιμίας

$$y_{1,j} + A_1 x + B_{1,j} \equiv 0 \bmod q \ , \ j \in \{1, 2, 3\} \tag{8.15}$$

ο ακέραιος A_1 λαμβάνει όλες τις τιμές στο διάστημα

$$(L_{1_{min}} + L_1, L_{1_{min}} + L_1 + l_1)$$

και κρατείται εκείνος ο οποίος ελαχιστοποιεί κάποια από τις ποσότητες $y_{1,j}$. Για το j αυτής της ποσότητας, επιλέγεται και η αντίστοιχη εξίσωση ισοτιμίας 8.15. Η διαδικασία αυτή, επαναλαμβάνεται και για τους υπόλοιπους ακέραιους A_2, A_3 (εάν $n = 3$) έτσι ώστε να συμπληρωθεί το σύστημα ισοτιμιών των n εξισώσεων.

Από το τελικό σύστημα των ισοτιμιών, αναμένεται (ανάλογα με τις επιλογές των τιμών L_i, l_i) η Ευκλείδεια νόρμα τους διανύσματος της λύσης, να είναι μικρότερη από το απαιτούμενο άνω φράγμα της.

8.3.2 Επίθεση σε Επιλεγμένο Κειμένο

Έστω ότι έχουμε το παρακάτω κείμενο M:

The Trump administration has stopped disclosing material information about the size and nature of the U.S. commitment to military action in Iraq and Syria, including the number of U.S. troops deployed in either country, in a bid to "surprise" the Islamic State with the number of US troops in the region the LA Times reports.

Αρχικά, θα επιλέξουμε μια ελλειπτική καμπύλη μεγάλης τάξης, θα δημιουργήσουμε με αυτήν το δημόσιο και το ιδιωτικό κλειδί μας και στη συνέχεια θα υπογράψουμε ξεχωριστά, τρία αυθαίρετα τμήματα του παραπάνω κειμένου.

Με την εντολή number(1,160,160) επιλέγουμε τον πρώτο αριθμό

$$p = 2^{160} + 291.$$

Με την εντολή elliptic_curve(1,0,0,2^160+291,0) παίρνουμε την ελλειπτική καμπύλη

$$E : y^2 \equiv x^3 + \underbrace{279536837310}_{a}\, x + \underbrace{12475593522979396813}_{b} \bmod \left(2^{160} + 291\right)$$

η οποία έχει τάξη

$$|E(\mathbb{Z}_p)| = 1461501637330902918203682645689535246225788841947.$$

Στη συνέχεια, επιλέγουμε $q = |E(\mathbb{Z}_p)|$ και χρησιμοποιώντας την εντολή points_generation(1,q,a,b,p), όπου q,a,b,p οι προηγούμενες τιμές, επιλέγουμε ένα τυχαίο σημείο της ελλειπτικής καμπύλης τάξης q, το P :

$$\begin{aligned} X_P &= 544619712320129078087170441367983800650160397449 \\ Y_P &= 618682169141419552202978351947104873532993550948 \end{aligned}$$

Με την εντολή numbers(1,160,160,0), επιλέγουμε το ιδιωτικό κλειδί τον 160-bits

$$A = 1165251305439503678027260487779494317936777590201.$$

Χρησιμοποιώντας την εντολή keys_generation(q,a,b,p,Xp,Yp,A) με τις τιμές των μεταβλητών που ορίστηκαν μέχρι τώρα, παίρνουμε το δημόσιο κλειδί Q :

$$\begin{aligned} X_Q &= 330918280029339895404982633723289724701055006495 \\ Y_Q &= 284743903801816987914876392784681939264296575704 \end{aligned}$$

Έστω τώρα ότι έχουμε στη διάθεσή μας τρία επιμέρους μηνύματα του αρχικού κειμένου

$$\begin{aligned} M_1 &= \text{The Trump administration} \\ M_2 &= \text{military action in Iraq} \\ M_3 &= \text{the LA Times reports} \end{aligned}$$

με τις αντίστοιχες συμπυκνώσεις τους

$$\begin{aligned} \mathcal{H}(M_1) &= 135086653280790040051634074976296178981107941172 4 \\ \mathcal{H}(M_2) &= 798990407035268294770813429236418959784817465834 \\ \mathcal{H}(M_3) &= 105268935728090729054323359201738228404177713783 4 \end{aligned}$$

Στη συνέχεια, εκτελώντας τρείς φορές ξεχωριστά για κάθε μήνυμα, την εντολή ECDSA_Signature(q,a,b,p,Xp,Yp,A,0,1,H) με τις τιμές των μεταβλητών όπως έχουν επιλεγεί μέχρι τώρα, λαμβάνουμε τα τρία τυχαία εφήμερα κλειδιά

$$\begin{aligned} k_1 &= 97239740413752308905742414963395794526897607590 3 \\ k_2 &= 54077009247664106480741142241479966515170764776 6 \\ k_3 &= 64931898456691285246543533703286771854886234095 8 \end{aligned}$$

τις υπογραφές

$$\begin{aligned} r_1 &= 120345874264146130148632654317196885971548423799 7 \\ r_2 &= 115614015024115058405582977509942646860468700687 3 \\ r_3 &= 52647400516754587949666342405677365801017249157 9 \\ s_1 &= 67389348977180926971311072364535105588699964401 3 \\ s_2 &= 10580218037908889877303451576612351522803401539 11 \\ s_3 &= 13120430129095032340715735796733377725372565953 77 \end{aligned}$$

και τον αριθμό

$$A^{-1} = 74.$$

Επειδή

$$\log\log q - 1 \approx 4.709$$

εφαρμόζουμε τον αλγόριθμο ECDSA-ATTACK-1 για $t = n = 3$. Με την εντολή numbers(1,0,0,x) παρατηρούμε ότι ο αριθμός A^{-1} έχει μέγεθος 7 bits και ο αριθμός A έχει μέγεθος 160 bits. Επομένως, η υλοποίηση της επίθεσής μας είναι προτιμότερο να γίνει με την επίλυση των συστημάτων της μορφής (8.3).

Εκτελώντας την εντολή $Attack\_Values(q, 3)$ παίρνουμε

$$\begin{aligned}
L_{1_{min}} &= 1099511627775 \\
L_{1_{max}} &= 2199023255551 \\
l_{1_{max}} &= 1099511627776 \\
L_{2_{min}} &= 2417851639229258349412350 \\
L_{2_{max}} &= 4835703278458516698824700 \\
l_{2_{max}} &= 4835703278457417187196925 \\
L_{3_{min}} &= 5316911983139663491615222273853659097 \\
L_{3_{max}} &= 10633823966279326983230444547707318194 \\
l_{3_{max}} &= 5316911983139663491615222273853659097
\end{aligned}$$

Για την εύρεση των βέλτιστων τιμών των ακεραίων A_i , $i = 1, \ldots, 3$ με τις αντίστοιχες εξισώσεις ισοτιμίας, θα εκτελέσουμε τρείς επαναλήψεις αναζήτησης, σε διαφορετικά διαστήματα τιμών σταθερού μήκους, με την εντολή

$$ECDSA\_ATTACK\_1$$

$$(q, a, b, p, X_p, Y_p, A, k_1, k_2, k_3, k, H_1, H_2, H_3, L_1, l_1, L_2, l_2, L_3, l_3)$$

Πρώτη Εκτέλεση Επίθεσης

Επιλέγουμε

$$\begin{aligned}
L_1 &= 1093693000000 &, \quad l_1 &= 500000 \\
L_2 &= 556256125226951713000000 &, \quad l_2 &= 500000 \\
L_3 &= 432062746460215536305306754500500000 &, \quad l_3 &= 500000
\end{aligned}$$

και έχουμε:

$$\begin{aligned}
y_{1,2} + A_1 x + B_{1,2} &\equiv 0 \bmod q \\
y_{2,2} + A_2 x + B_{2,2} &\equiv 0 \bmod q \\
y_{3,1} + A_3 x + B_{3,1} &\equiv 0 \bmod q
\end{aligned}$$

με λύση $\mathbf{v} = (x, y_{1,2}, y_{2,2}, y_{3,1})$ όπου

$$\begin{aligned}
x &= 74 \\
y_{1,2} &= 228294776460572766928839057898274204570286 0 \\
y_{2,2} &= 92921604714125369246147 1991 \\
y_{3,1} &= 341157971296584694420841004963946071537376 \\
A_1 &= 2193205071441 \\
A_2 &= 2974107764456210062737971 \\
A_3 &= 57489747295998790279205290283545891 22 \\
B_{1,2} &= 14614993543831383124760133572989561011865678524 53 \\
B_{2,2} &= 14615016373309029182025333456678242329886847601 02 \\
B_{3,1} &= 14615012957475074916285971767824111341814777095 43
\end{aligned}$$

και

$$\begin{aligned}
\lfloor \|\mathbf{v}\| \rfloor &= 230829791346296768665552945311023522951520 4 \\
\left\lfloor \tfrac{q^{3/4}}{16} \right\rfloor &= 83076749736557242056487848028963423
\end{aligned}$$

Δεύτερη Εκτέλεση Επίθεσης

Επιλέγουμε

$$\begin{aligned}
L_1 &= 1093693500000 &, \quad l_1 &= 500000 \\
L_2 &= 556256125226951713500000 &, \quad l_2 &= 500000 \\
L_3 &= 432062746460215536305306754501000000 &, \quad l_3 &= 500000
\end{aligned}$$

και έχουμε:

$$\begin{aligned}
y_{1,2} + A_1 x + B_{1,2} &\equiv 0 \bmod q \\
y_{2,2} + A_2 x + B_{2,2} &\equiv 0 \bmod q \\
y_{3,3} + A_3 x + B_{3,3} &\equiv 0 \bmod q
\end{aligned}$$

με λύση $\mathbf{v} = (x, y_{1,2}, y_{2,2}, y_{3,3})$ όπου

$$\begin{aligned}
x &= 74 \\
y_{1,2} &= 1104420231612720246988349635541118611919 75 \\
y_{2,2} &= 772900766733407038681779418560259568074469 \\
y_{3,3} &= 256347472845201069497 \\
A_1 &= 2193205286345 \\
A_2 &= 2974107764456210063334242 \\
A_3 &= 57489747295998790279205290283550746 44 \\
B_{1,2} &= 14615015268888797569316579468545715298167364604 42 \\
B_{2,2} &= 14615008644301361847964238799355469264215340335 70 \\
B_{3,3} &= 14615016369054787882132915973670686252823122487 94
\end{aligned}$$

και

$$\begin{array}{rcl} \lfloor \|\mathbf{v}\| \rfloor & = & 78075158385817151466691594635251814667665 9 \\ \left\lfloor \frac{q^{3/4}}{16} \right\rfloor & = & 830767497365572420564878480289634 23 \end{array}$$

Τρίτη Εκτέλεση Επίθεσης

Επιλέγουμε

$$\begin{array}{rclcrcl} L_1 & = & 1093694000000 & , & l_1 & = & 500000 \\ L_2 & = & 556256125226951714000000 & , & l_2 & = & 500000 \\ L_3 & = & 432062746460215536305306754501500000 & , & l_3 & = & 500000 \end{array}$$

και έχουμε:

$$\begin{array}{rcl} y_{1,1} + A_1 x + B_{1,1} & \equiv & 0 \bmod q \\ y_{2,3} + A_2 x + B_{2,3} & \equiv & 0 \bmod q \\ y_{3,3} + A_3 x + B_{3,3} & \equiv & 0 \bmod q \end{array}$$

με λύση $\mathbf{v} = (x, y_{1,1}, y_{2,3}, y_{3,3})$ όπου

$$\begin{array}{rcl} x & = & 74 \\ y_{1,1} & = & 13500054495005086060101 \\ y_{2,3} & = & 113275550477073834129517230398890167642 2296 \\ y_{3,3} & = & 154915371710299651682695630242022891233 7788 \\ A_1 & = & 2193205956088 \\ A_2 & = & 29741077644562100635614 08 \\ A_3 & = & 57489747295998790279205290283555928 21 \\ B_{1,1} & = & 14615016373309029182036826321894805889234620313 34 \\ B_{2,3} & = & 14615005045753981474651212665426614977794088754 59 \\ B_{3,3} & = & 14615000877517616852167747706671136778985626354 05 \end{array}$$

και

$$\begin{array}{rcl} \lfloor \|\mathbf{v}\| \rfloor & = & 191911757659671315299900049521099193788003 9 \\ \left\lfloor \frac{q^{3/4}}{16} \right\rfloor & = & 830767497365572420564878480289634 23 \end{array}$$

Επιλογή Εξισώσεων Ισοτιμιών

Από τις τρείς προηγούμενες εκτελέσεις των επιθέσεων, παρατηρούμε ότι οι βέλτιστες επιλογές των A_i , $i = 1, \ldots, 3$ είναι οι τιμές:

$$\begin{aligned} A_1 &= 2193205956088 \text{ από την } 3^{\eta} \text{ εκτέλεση} \\ A_2 &= 297410776445621006273797 1 \text{ από την } 1^{\eta} \text{ εκτέλεση} \\ A_3 &= 57489747295998790279205290283550746 44 \text{ από την } 2^{\eta} \text{ εκτέλεση.} \end{aligned}$$

Έτσι αν αρχικά είχαμε επιλέξει

$$\begin{aligned} L_1 &= 1093694000000 & , \quad l_1 &= 500000 \\ L_2 &= 556256125226951713000000 & , \quad l_2 &= 500000 \\ L_3 &= 432062746460215536305306754501000000 & , \quad l_3 &= 500000 \end{aligned}$$

η εκτέλεση της επίθεσης σε αυτά τα διαστήματα θα έδινε

$$\begin{aligned} C_1 &= 213030503248746234536915869628025704446053826214 \\ C_2 &= 233867172665830015951489085506979357705827758845 \\ C_3 &= 998102630670735082546729021845931413804021975191 \\ D_1 &= 111822615064392235817302780406148408710247748 3829 \\ D_2 &= 288201819818772407595521943350590211029146 31391 \\ D_3 &= 936944934259200678506883112683435302133365689076 \end{aligned}$$

με βέλτιστο σύστημα ισοτιμιών

$$\begin{aligned} y_{11} + A_1 x + B_{11} &\equiv 0 \bmod q \\ y_{22} + A_2 x + B_{22} &\equiv 0 \bmod q \\ y_{33} + A_3 x + B_{33} &\equiv 0 \bmod q \end{aligned}$$

λύση $v_1 = (x, y_{11}, y_{22}, y_{33})$, όπου

$$\begin{aligned} x &= 74 \\ y_{11} &= 13500054495005086060101 \\ y_{22} &= 92921604714125369246147 1991 \\ y_{33} &= 256347472845201069497 \\ A_1 &= 2193205956088 \\ A_2 &= 297410776445621006273797 1 \\ A_3 &= 57489747295998790279205290283550746 44 \\ B_{11} &= 14615016373309029182036826321894805889234620313 34 \\ B_{22} &= 14615016373309029182025333456678242329886847601 02 \\ B_{33} &= 14615016369054787882132915973670686252823122487 94 \end{aligned}$$

και

$$\begin{aligned} \lfloor \|\mathbf{v}\| \rfloor &= 92921604723935638125285 6089 \\ \left\lfloor \frac{q^{3/4}}{16} \right\rfloor &= 83076749736557242056487848028963423 \end{aligned}$$

Συνεπώς ισχύει:

$$\sqrt{x^2 + y_{1,1}^2 + y_{2,2}^2 + y_{3,3}^2} < \frac{q^{3/4}}{16}$$

οπότε από το Θεώρημα 7.2 αποκαλύπτεται η τιμή $A^{-1} \bmod q$ και κατά συνέπεια το ιδιωτικό κλειδί A του σχήματος ψηφιακής υπογραφής ECDSA.

Παραρτήματα

Παράρτημα Α΄

Κώδικας Αλγόριθμου ECDSA

```
def numbers(n,a,b,x):
    if (n>=1):
     prime=[0]*n;
     for i in range(n):
      if ((a<=b) and (0<a) and (0<b) and (x==0)):
       r=randrange(a,b+1);
       prime[i]=next_prime(randrange(2^(r-1),2^r));
      elif ((n==1) and (a==0) and (b==0) and (x!=0)):
       return int(math.log(x, 2)) + 1
      else:
       prime=0;
    else:
     prime=0;
    return prime

def elliptic_curve(n,a,b,p,prt):
    F=Zmod(p);
    if (is_prime(p)):
     if ((((4*a^3+27*b^2)%p)!=0) and (p>4)):
      if ((n==1) and (prt==0)):
       E=EllipticCurve(F,[a,b]);
       return [E,E.order()]
      else:
       return 0
     elif (((a==0) and (b==0)) and (n>=1) and (p>4)):
      lE=[0]*(2*n);
      for i in range(0,2*n,2):
       s=floor(log(p,2))+1;
```

```
       r=randrange(2,s+1);
       a=randrange(2^(r-1),2^r);
       r=randrange(2,s+1);
       b=randrange(2^(r-1),2^r);
       while (((4*a^3+27*b^2)%p)==0):
        r=randrange(2,s+1);
        a=randrange(2^(r-1),2^r);
        r=randrange(2,s+1);
        b=randrange(2^(r-1),2^r);
       E=EllipticCurve(F,[a,b]);
       while ((is_prime(E.order()))==False):
        r=randrange(2,s+1);
        a=randrange(2^(r-1),2^r);
        r=randrange(2,s+1);
        b=randrange(2^(r-1),2^r);
        while (((4*a^3+27*b^2)%p)==0):
         r=randrange(2,s+1);
         a=randrange(2^(r-1),2^r);
         r=randrange(2,s+1);
         b=randrange(2^(r-1),2^r);
        E=EllipticCurve(F,[a,b]);
       lE[i]=E;
       lE[i+1]=E.order();
      if (prt==0):
       return lE
      elif (prt==1):
       for i in range(0,2*n,2):
        print lE[i],"  #E =",lE[i+1]
      else:
       return 0
     else:
      return 0
    else:
     return 0

def prime_divisors(x,n):
    if (x>0):
     lx=len(factor(x));
     pd=[0]*lx;
     if (n==0):
      for i in range(lx):
```

```
       pd[i]=factor(x)[i][0];
      return pd
     elif (n==1):
      pd[0]=factor(x)[lx-1][0];
      return pd[0]
     else:
      return 0
    else:
     return 0

def points_generation(n,q,a,b,p):
    E=elliptic_curve(1,a,b,p,0);
    if (E!=0):
     if ((q==0) and (a==0) and (b==0)):
      q=E[1];
     if (n>=1):
      lP=[0]*(n+2);
      lP[0]=E[0];
      lP[1]=E[1];
      for i in range(n):
       i=i+1;
       P=E[0].random_point();
       PO=P.order();
       c=0;
       while ((PO!=q) and (c<1000)):
        P=E[0].random_point();
        PO=P.order();
        c=c+1;
       if (c<1000):
        lP[i+1]=P;
       else:
        return 0
      return lP;
     else:
      return 0
    else:
     return 0

def keys_generation(q,a,b,p,Xp,Yp,A):
    lP=points_generation(1,q,a,b,p);
    if (lP!=0):
```

```
     if ((Xp==0) and (Yp==0)):
      P=lP[2];
      PO=P.order();
      if (A==0):
       A=randrange(0,PO-1);
       Q=A*P;
       return (lP[0],lP[1],P,Q,A)
      elif ((A>0) and (A<PO-1)):
       Q=A*P;
       return (lP[0],lP[1],P,Q,A)
      else:
       return 0
     elif ((a!=0) | (b!=0)):
      if ((abs(Yp^2-Xp^3-a*Xp-b)%p)==0):
       E=lP[0];
       P=E(Xp,Yp);
       PO=P.order();
       if (A==0):
        A=randrange(0,PO-1);
        Q=A*P;
        return (lP[0],lP[1],P,Q,A)
       elif ((A>0) and (A<=PO-1)):
        Q=A*P;
        return (lP[0],lP[1],P,Q,A)
       else:
        return 0
      else:
       return 0
     else:
      return 0
    else:
     return 0

def message_representation(prt):
    if ((prt==1) | (prt==0)):
     m = raw_input("Type your message here:");
     import hashlib
     M=str(m);
     from sage.crypto.util import ascii_to_bin
     t=ascii_to_bin(str(M));
     z=int(str(t),2)+0;
```

```
      x=hex(int('1'+str(t), 2))[3:];
      r=hashlib.sha224(str(M)).hexdigest();
      h=bin(int('1'+r, 16))[3:];
      c=int(str(h),2)+0;
      if (prt==0):
       return (t,z,x,h,c,r)
      if (prt==1):
       print "  Message in Binary:",t;
       print "  Message in Decimal:",z;
       print "  Message in Hexadical:",x;
       print "  SHA224 in Binary:",h;
       print "  SHA224 in Decimal:",c;
       print "  SHA224 in Hexadical:",r;
     else:
      return 0

def ECDSA_Signature(q,a,b,p,Xp,Yp,A,k,prt,H):
     key=keys_generation(q,a,b,p,Xp,Yp,A);
     if (key!=0):
      if ((k>=0) and (k<key[2].order())):
       P=key[2];
       t=P.order();
       if (H==0):
        m=message_representation(0);
        m2=int(m[4]*float(2^(-224+numbers(1,0,0,t))));
       if (H!=0):
        m2=H;
       if (k==0):
        k=randrange(1,t);
       kP=k*P;
       kk=inverse_mod(k,t);
       AA=inverse_mod(key[4],t);
       r=mod(kP[0],t);
       s=mod((kk)*(m2+key[4]*r),t);
       if (prt==1):
        if (H==0):
         print "Message M =",m[1];
        print " ",key[0];
        print "  #E =",key[1];
        print "  q =",t;
        print "  Public Key:  ","P =",key[2],", Q =",key[3];
```

```
       print "  Private Key: ","A =",key[4],", k =",k;
       print "  k^(-1) =",kk;
       print "  A^(-1) =",AA;
       print "  H(M): ",m2;
       print "ECDSA_Signature (r,s) =","(",r,",",s,")";
      elif ((prt==0) and (H==0)):
       return (m[1],key[0],key[1],t,key[2],key[3],key[4],k,kk,AA,m2,r,s)
      elif ((prt==0) and (H!=0)):
       return (0,key[0],key[1],t,key[2],key[3],key[4],k,kk,AA,m2,r,s)
      else:
       return 0
     else:
      return 0
    else:
     return 0

def Message_Authenticity(q,a,b,p,Xp,Yp,Xq,Yq,r,s,H):
    if ((0<r) and (r<q) and (0<s) and (s<q)):
     E=elliptic_curve(1,a,b,p,0)[0];
     if (H==0):
      m=message_representation(0);
      m2=int(m[4]*float(2^(-224+numbers(1,0,0,q))));
     if (H!=0):
      m2=H;
     w=inverse_mod(s,q);
     u1=mod(m2*int(w),q);
     u2=mod(r*int(w),q);
     P=E(Xp,Yp);
     Q=E(Xq,Yq);
     R=(int(u1)*P)+(int(u2)*Q);
     v=mod(R[0],q);
     if (v==r):
      return  "v=r"
     else:
      return  "v!=r"
    else:
     return  "message is not genuine"
```

Παράρτημα Β΄

Κώδικας Αλγόριθμων Επίθεσης

```
def Attack_Values(q,n):
    L1min=floor(q^(1/(n+1)));
    L2min=floor(2*q^(2/(n+1)));
    L3min=floor(4*q^(3/(n+1)));
    L1max=floor(2*q^(1/(n+1)));
    L2max=floor(4*q^(2/(n+1)));
    L3max=floor(8*q^(3/(n+1)));
    l1=L1max-L1min;
    l2=L2max-L2min;
    l3=L3max-L3min;
    print "L1 min value     = ",L1min;
    print "L1 max value     = ",L1max;
    print "l1 = L1max-L1min = ",L1max-L1min;
    print ""
    print "L2 min value     = ",L2min;
    print "L2 max value     = ",L2max;
    print "l2 = L2max-L2min = ",L2max-L1min;
    print ""
    print "L3 min value     = ",L3min;
    print "L3 max value     = ",L3max;
    print "l3 = L3max-L3min = ",L3max-L3min;

def ECDSA_ATTACK_1(q,a,b,p,Xp,Yp,A,k1,k2,k3,k,H1,H2,H3,L1,l1,L2,l2,L3,l3):
    if ((k!=1) and (k!=2)):
     return 0
    n=0;
    if ((k1!=0) and (k2!=0) and (k3==0)):
```

```
 n=2;
elif ((k1!=0) and (k2!=0) and (k3!=0)):
 n=3;
else:
 return 0
if (n!=0):
 S1=ECDSA_Signature(q,a,b,p,Xp,Yp,A,k1,0,H1);
 C1=mod((-S1[11])*(inverse_mod(int(S1[12]),S1[3])),S1[3]);
 D1=mod((-(inverse_mod(int(S1[12]),S1[3])))*S1[10],S1[3]);
 CC1=inverse_mod(int(C1),S1[3]);
 DD1=inverse_mod(int(D1),S1[3]);

 S2=ECDSA_Signature(q,a,b,p,Xp,Yp,A,k2,0,H2);
 C2=mod((-S2[11])*(inverse_mod(int(S2[12]),S2[3])),S2[3]);
 D2=mod((-(inverse_mod(int(S2[12]),S2[3])))*S2[10],S2[3]);
 CC2=inverse_mod(int(C2),S2[3]);
 DD2=inverse_mod(int(D2),S2[3]);

 if ((n==2) and (k==2)):
  print "n=t=2"
  print ""
  print "C1  =",C1
  print "C2  =",C2
  print "D1  =",D1
  print "D2  =",D2
  print ""
  print "Optimal System"
  print ""

  m1a=2^333;
  A1a=int(floor((S1[3])^(1/4)))+L1;
  if (floor(A1a+l1)>floor(2*(S1[3])^(1/4))):
   print "1.    L1 or l1 wrong value"
   print ""
   y11=0;
   a1=0;
   B11=0;
  else:
   for i in range(l1):
    T=A1a+i;
    if (mod(int(k1*S1[9]*T*DD1),S1[3])<m1a):
```

```
   m1a=mod(int(k1*S1[9]*T*DD1),S1[3]);
   a1=T;
 y11=m1a;
 B11=mod(int(C1*a1*DD1),S1[3]);

m1b=2^333;
A1b=int(floor((S1[3])^(1/4)))+L1;
if (floor(A1b+l1)>floor(2*(S1[3])^(1/4))):
 y12=0;
 a2=0;
 B12=0;
else:
 A1b=int(floor((S2[3])^(1/4))+L1);
 for i in range(l1):
  T=A1b+i;
  if (mod(int(k2*S2[9]*T*DD2),S2[3])<m1b):
   m1b=mod(int(k2*S2[9]*T*DD2),S2[3]);
   a2=T;
 y12=m1b;
 B12=mod(int(C2*a2*DD2),S2[3]);

y1=min(y11,y12);
if ((y1==y11) and (y1!=0)):
 print "1.    y11+A1*x+B11=0 mod q"
if ((y1==y12) and (y1!=0)):
 print "1.    y12+A1*x+B12=0 mod q"

m2a=2^333;
A2a=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2a+l2)>floor(4*(S1[3])^(2/4))):
 print "2.    L2 or l2 wrong value"
 print ""
 y21=0;
 b1=0;
 B21=0;
else:
 for i in range(l2):
  T=A2a+i;
  if (mod(int(k1*S1[9]*T*DD1),S1[3])<m2a):
   m2a=mod(int(k1*S1[9]*T*DD1),S1[3]);
   b1=T;
```

```
 y21=m2a;
 B21=mod(int(C1*b1*DD1),S1[3]);

m2b=2^333;
A2b=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2b+l2)>floor(4*(S1[3])^(2/4))):
 y22=0;
 b2=0;
 B22=0;
else:
 for i in range(l2):
  T=A2b+i;
  if (mod(int(k2*S2[9]*T*DD2),S2[3])<m2b):
   m2b=mod(int(k2*S2[9]*T*DD2),S2[3]);
   b2=T;
 y22=m2b;
 B22=mod(int(C2*b2*DD2),S2[3]);

y2=min(y21,y22);
if ((y2==y21) and (y2!=0)):
 print "2.    y21+A2*x+B21=0 mod q"
if ((y2==y22) and (y2!=0)):
 print "2.    y22+A2*x+B22=0 mod q"

print ""
print "q   =",S1[3]
print "x   =",S1[9]
if ((y1==y11) and (y1!=0)):
 print "y11 =",y11
if ((y1==y12) and (y1!=0)) :
 print "y12 =",y12
if ((y2==y21) and (y2!=0)) :
 print "y21 =",y21
if ((y2==y22) and (y2!=0)):
 print "y22 =",y22
if ((y1==y11) and (y1!=0)):
 print "A1  =",a1
if ((y1==y12) and (y1!=0)):
 print "A1  =",a2
if ((y2==y21) and (y2!=0)):
 print "A2  =",b1
```

```
 if ((y2==y22) and (y2!=0)):
  print "A2  =",b2
 if ((y1==y11) and (y1!=0)) :
  print "B11 =",B11
 if ((y1==y12) and (y1!=0)):
  print "B12 =",B12
 if ((y2==y21) and (y2!=0)):
  print "B21 =",B21
 if ((y2==y22)and (y2!=0)):
  print "B22 =",B22
 print ""
 print "q^(3/4)/16  =",floor(((S1[3])^(2/3))/16)
 print "|v|         =",floor(((S1[9])^2+int(y1)^2+int(y2)^2)^(1/2))

if ((n==3) and (k==2)):
 S3=ECDSA_Signature(q,a,b,p,Xp,Yp,A,k3,0,H3);
 C3=mod((-S3[11])*(inverse_mod(int(S3[12]),S3[3])),S3[3]);
 D3=mod((-(inverse_mod(int(S3[12]),S3[3])))*S3[10],S3[3]);
 CC3=inverse_mod(int(C3),S3[3]);
 DD3=inverse_mod(int(D3),S3[3]);

 print "n=t=3"
 print ""
 print "C1  =",C1
 print "C2  =",C2
 print "C3  =",C3
 print "D1  =",D1
 print "D2  =",D2
 print "D3  =",D3
 print ""
 print "Optimal System"
 print ""

if (n==3):
 if (k==2):
  m1a=2^333;
  A1a=int(floor((S1[3])^(1/4)))+L1;
  if (floor(A1a+l1)>floor(2*(S1[3])^(1/4))):
   print "1.    L1 or l1 wrong value"
   print ""
   y11=0;
```

```
 a1=0;
 B11=0;
else:
 for i in range(l1):
  T=A1a+i;
  if (mod(int(k1*S1[9]*T*DD1),S1[3])<m1a):
   m1a=mod(int(k1*S1[9]*T*DD1),S1[3]);
   a1=T;
 y11=m1a;
 B11=mod(int(C1*a1*DD1),S1[3]);

m1b=2^333;
A1b=int(floor((S1[3])^(1/4)))+L1;
if (floor(A1b+l1)>floor(2*(S1[3])^(1/4))):
 y12=0;
 a2=0;
 B12=0;
else:
 for i in range(l1):
  T=A1b+i;
  if (mod(int(k2*S2[9]*T*DD2),S2[3])<m1b):
   m1b=mod(int(k2*S2[9]*T*DD2),S2[3]);
   a2=T;
 y12=m1b;
 B12=mod(int(C2*a2*DD2),S2[3]);

m1c=2^333;
A1c=int(floor((S1[3])^(1/4)))+L1;
if (floor(A1c+l1)>floor(2*(S1[3])^(1/4))):
 y13=0;
 a3=0;
 B13=0;
else:
 for i in range(l1):
  T=A1c+i;
  if (mod(int(k3*S3[9]*T*DD3),S3[3])<m1c):
   m1c=mod(int(k3*S3[9]*T*DD3),S3[3]);
   a3=T;
 y13=m1c;
 B13=mod(int(C3*a3*DD3),S3[3]);
```

```
y1=min(y11,y12,y13);
if ((y1==y11) and (y1!=0)):
 print "1.    y11+A1*x+B11=0 mod q"
if ((y1==y12) and (y1!=0)):
 print "1.    y12+A1*x+B12=0 mod q"
if ((y1==y13) and (y1!=0)):
 print "1.    y13+A1*x+B13=0 mod q"

m2a=2^333;
A2a=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2a+l2)>floor(4*(S1[3])^(2/4))):
 print "2.    L2 or l2 wrong value"
 print ""
 y21=0;
 b1=0;
 B21=0;
else:
 for i in range(l2):
  T=A2a+i;
  if (mod(int(k1*S1[9]*T*DD1),S1[3])<m2a):
   m2a=mod(int(k1*S1[9]*T*DD1),S1[3]);
   b1=T;
 y21=m2a;
 B21=mod(int(C1*b1*DD1),S1[3]);

m2b=2^333;
A2b=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2b+l2)>floor(4*(S1[3])^(2/4))):
 y22=0;
 b2=0;
 B22=0;
else:
 for i in range(l2):
  T=A2b+i;
  if (mod(int(k2*S2[9]*T*DD2),S2[3])<m2b):
   m2b=mod(int(k2*S2[9]*T*DD2),S2[3]);
   b2=T;
 y22=m2b;
 B22=mod(int(C2*b2*DD2),S2[3]);

m2c=2^333;
```

```
A2c=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2c+l2)>floor(4*(S1[3])^(2/4))):
 y23=0;
 b3=0;
 B23=0;
else:
 for i in range(l2):
  T=A2c+i;
  if (mod(int(k3*S3[9]*T*DD3),S2[3])<m2c):
   m2c=mod(int(k3*S3[9]*T*DD3),S2[3]);
   b3=T;
 y23=m2c;
 B23=mod(int(C3*b3*DD3),S3[3]);

y2=min(y21,y22,y23);
if ((y2==y21) and (y2!=0)):
 print "2.    y21+A2*x+B21=0 mod q"
if ((y2==y22) and (y2!=0)):
 print "2.    y22+A2*x+B22=0 mod q"
if ((y2==y23) and (y2!=0)):
 print "2.    y23+A2*x+B23=0 mod q"

m3a=2^333;
A3a=int(floor(4*(S1[3])^(3/4)))+L3;
if (floor(A3a+l3)>floor(8*(S1[3])^(3/4))):
 print "3.    L3 or l3 wrong value"
 print ""
 y31=0;
 c1=0;
 B31=0;
else:
 for i in range(l3):
  T=A3a+i;
  if (mod(int(k1*S1[9]*T*DD1),S1[3])<m3a):
   m3a=mod(int(k1*S1[9]*T*DD1),S1[3]);
   c1=T;
 y31=m3a;
 B31=mod(int(C1*c1*DD1),S1[3]);

m3b=2^333;
A3b=int(floor(4*(S1[3])^(3/4)))+L3;
```

```
if (floor(A3b+l3)>floor(8*(S1[3])^(3/4))):
 y32=0;
 c2=0;
 B32=0;
else:
 for i in range(l3):
  T=A3b+i;
  if (mod(int(k2*S2[9]*T*DD2),S2[3])<m3b):
   m3b=mod(int(k2*S2[9]*T*DD2),S2[3]);
   c2=T;
 y32=m3b;
 B32=mod(int(C2*c2*DD2),S2[3]);

m3c=2^333;
A3c=int(floor(4*(S1[3])^(3/4)))+L3;
if (floor(A3c+l3)>floor(8*(S1[3])^(3/4))):
 y33=0;
 c3=0;
 B33=0;
else:
 for i in range(l3):
  T=A3c+i;
  if (mod(int(k3*S3[9]*T*DD3),S2[3])<m3c):
   m3c=mod(int(k3*S3[9]*T*DD3),S2[3]);
   c3=T;
 y33=m3c;
 B33=mod(int(C3*c3*DD3),S3[3]);

y3=min(y31,y32,y33);
if ((y3==y31) and (y3!=0)):
 print "3.    y31+A3*x+B31=0 mod q"
if ((y3==y32) and (y3!=0)):
 print "3.    y32+A3*x+B32=0 mod q"
if ((y3==y33) and (y3!=0)):
 print "3.    y33+A3*x+B33=0 mod q"

print ""
print "q   =",S1[3]
print "x   =",S1[9]
if ((y1==y11) and (y1!=0)):
 print "y11 =",y11
```

```
if ((y1==y12) and (y1!=0)):
 print "y12 =",y12
if ((y1==y13) and (y1!=0)):
 print "y13 =",y13
if ((y2==y21) and (y2!=0)):
 print "y21 =",y21
if ((y2==y22) and (y2!=0)):
 print "y22 =",y22
if ((y2==y23) and (y2!=0)):
 print "y23 =",y23
if ((y3==y31) and (y3!=0)):
 print "y31 =",y31
if ((y3==y32) and (y3!=0)):
 print "y32 =",y32
if ((y3==y33) and (y3!=0)):
 print "y33 =",y33
if ((y1==y11) and (y1!=0)):
 print "A1  =",a1
if ((y1==y12) and (y1!=0)):
 print "A1  =",a2
if ((y1==y13) and (y1!=0)):
 print "A1  =",a3
if ((y2==y21) and (y2!=0)):
 print "A2  =",b1
if ((y2==y22) and (y2!=0)):
 print "A2  =",b2
if ((y2==y23) and (y2!=0)):
 print "A2  =",b3
if ((y3==y31) and (y3!=0)):
 print "A3  =",c1
if ((y3==y32) and (y3!=0)):
 print "A3  =",c2
if ((y3==y33) and (y3!=0)):
 print "A3 =",c3
if ((y1==y11) and (y1!=0)):
 print "B11 =",B11
if ((y1==y12) and (y1!=0)):
 print "B12 =",B12
if ((y1==y13) and (y1!=0)):
 print "B13 =",B13
if ((y2==y21) and (y2!=0)):
```

```
         print "B21 =",B21
        if ((y2==y22) and (y2!=0)):
         print "B22 =",B22
        if ((y2==y23) and (y2!=0)):
         print "B23 =",B23
        if ((y3==y31) and (y3!=0)):
         print "B31 =",B31
        if ((y3==y32) and (y3!=0)):
         print "B32 =",B32
        if ((y3==y33) and (y3!=0)):
         print "B33 =",B33
        print ""
        print "q^(3/4)/16  =",floor(((S1[3])^(3/4))/16)
        print "|v|         =",
floor(((S1[9])^2+int(y1)^2+int(y2)^2+int(y3)^2)^(1/2))

     if ((n==2) and (k==1)):
      print "n=t=2"
      print ""
      print "C1  =",C1
      print "C2  =",C2
      print "D1  =",D1
      print "D2  =",D2
      print ""
      print "Optimal System"
      print ""

      m1a=2^333;
      A1a=int(floor((S1[3])^(1/4)))+L1;
      if (floor(A1a+l1)>floor(2*(S1[3])^(1/4))):
       print "1.    L1 or l1 wrong value"
       print ""
       y11=0;
       a1=0;
       B11=0;
      else:
       for i in range(l1):
        T=A1a+i;
        if (mod(int(k1*T*CC1),S1[3])<m1a):
         m1a=mod(int(k1*T*CC1),S1[3]);
         a1=T;
```

```
 y11=m1a;
 B11=mod(int(D1*a1*CC1),S1[3]);

m1b=2^333;
A1b=int(floor((S1[3])^(1/4)))+L1;
if (floor(A1b+l1)>floor(2*(S1[3])^(1/4))):
 y12=0;
 a2=0;
 B12=0;
else:
 A1b=int(floor((S2[3])^(1/4))+L1);
 for i in range(l1):
  T=A1b+i;
  if (mod(int(k2*T*CC2),S2[3])<m1b):
   m1b=mod(int(k2*T*CC2),S2[3]);
   a2=T;
 y12=m1b;
 B12=mod(int(D2*a2*CC2),S2[3]);

y1=min(y11,y12);
if ((y1==y11) and (y1!=0)):
 print "1.    y11+A1*x+B11=0 mod q"
if ((y1==y12) and (y1!=0)):
 print "1.    y12+A1*x+B12=0 mod q"

m2a=2^333;
A2a=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2a+l2)>floor(4*(S1[3])^(2/4))):
 print "2.    L2 or l2 wrong value"
 print ""
 y21=0;
 b1=0;
 B21=0;
else:
 for i in range(l2):
  T=A2a+i;
  if (mod(int(k1*T*CC1),S1[3])<m2a):
   m2a=mod(int(k1*T*CC1),S1[3]);
   b1=T;
 y21=m2a;
 B21=mod(int(D1*b1*CC1),S1[3]);
```

```
m2b=2^333;
A2b=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2b+l2)>floor(4*(S1[3])^(2/4))):
 y22=0;
 b2=0;
 B22=0;
else:
 for i in range(l2):
  T=A2b+i;
  if (mod(int(k2*T*CC2),S2[3])<m2b):
   m2b=mod(int(k2*T*CC2),S2[3]);
   b2=T;
 y22=m2b;
 B22=mod(int(D2*b2*CC2),S2[3]);

y2=min(y21,y22);
if ((y2==y21) and (y2!=0)):
 print "2.    y21+A2*x+B21=0 mod q"
if ((y2==y22) and (y2!=0)):
 print "2.    y22+A2*x+B22=0 mod q"

print ""
print "q   =",S1[3]
print "x   =",A
if ((y1==y11) and (y1!=0)):
 print "y11 =",y11
if ((y1==y12) and (y1!=0)) :
 print "y12 =",y12
if ((y2==y21) and (y2!=0)) :
 print "y21 =",y21
if ((y2==y22) and (y2!=0)):
 print "y22 =",y22
if ((y1==y11) and (y1!=0)):
 print "A1  =",a1
if ((y1==y12) and (y1!=0)):
 print "A1  =",a2
if ((y2==y21) and (y2!=0)):
 print "A2  =",b1
if ((y2==y22) and (y2!=0)):
 print "A2  =",b2
```

```
      if ((y1==y11) and (y1!=0)) :
       print "B11 =",B11
      if ((y1==y12) and (y1!=0)):
       print "B12 =",B12
      if ((y2==y21) and (y2!=0)):
       print "B21 =",B21
      if ((y2==y22)and (y2!=0)):
       print "B22 =",B22
      print ""
      print "q^(3/4)/16  =",floor(((S1[3])^(2/3))/16)
      print "|v|         =",
floor((int(A)^2+int(y1)^2+int(y2)^2)^(1/2))

     if ((n==3) and (k==1)):
      S3=ECDSA_Signature(q,a,b,p,Xp,Yp,A,k3,0,H3);
      C3=mod((-S3[11])*(inverse_mod(int(S3[12]),S3[3])),S3[3]);
      D3=mod((-(inverse_mod(int(S3[12]),S3[3])))*S3[10],S3[3]);
      CC3=inverse_mod(int(C3),S3[3]);
      DD3=inverse_mod(int(D3),S3[3]);

      print "n=t=3"
      print ""
      print "C1  =",C1
      print "C2  =",C2
      print "C3  =",C3
      print "D1  =",D1
      print "D2  =",D2
      print "D3  =",D3
      print ""
      print "Optimal System"
      print ""

     if (n==3):
      if (k==1):
       m1a=2^333;
       A1a=int(floor((S1[3])^(1/4)))+L1;
       if (floor(A1a+l1)>floor(2*(S1[3])^(1/4))):
        print "1.    L1 or l1 wrong value"
        print ""
        y11=0;
        a1=0;
```

```
 B11=0;
else:
 for i in range(l1):
  T=A1a+i;
  if (mod(int(k1*T*CC1),S1[3])<m1a):
   m1a=mod(int(k1*T*CC1),S1[3]);
   a1=T;
 y11=m1a;
 B11=mod(int(D1*a1*CC1),S1[3]);

m1b=2^333;
A1b=int(floor((S1[3])^(1/4)))+L1;
if (floor(A1b+l1)>floor(2*(S1[3])^(1/4))):
 y12=0;
 a2=0;
 B12=0;
else:
 for i in range(l1):
  T=A1b+i;
  if (mod(int(k2*T*CC2),S2[3])<m1b):
   m1b=mod(int(k2*T*CC2),S2[3]);
   a2=T;
 y12=m1b;
 B12=mod(int(D2*a2*CC2),S2[3]);

m1c=2^333;
A1c=int(floor((S1[3])^(1/4)))+L1;
if (floor(A1c+l1)>floor(2*(S1[3])^(1/4))):
 y13=0;
 a3=0;
 B13=0;
else:
 for i in range(l1):
  T=A1c+i;
  if (mod(int(k3*T*CC3),S3[3])<m1c):
   m1c=mod(int(k3*T*CC3),S3[3]);
   a3=T;
 y13=m1c;
 B13=mod(int(D3*a3*CC3),S3[3]);

y1=min(y11,y12,y13);
```

```
if ((y1==y11) and (y1!=0)):
 print "1.    y11+A1*x+B11=0 mod q"
if ((y1==y12) and (y1!=0)):
 print "1.    y12+A1*x+B12=0 mod q"
if ((y1==y13) and (y1!=0)):
 print "1.    y13+A1*x+B13=0 mod q"

m2a=2^333;
A2a=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2a+l2)>floor(4*(S1[3])^(2/4))):
 print "2.    L2 or l2 wrong value"
 print ""
 y21=0;
 b1=0;
 B21=0;
else:
 for i in range(l2):
  T=A2a+i;
  if (mod(int(k1*T*CC1),S1[3])<m2a):
   m2a=mod(int(k1*T*CC1),S1[3]);
   b1=T;
 y21=m2a;
 B21=mod(int(D1*b1*CC1),S1[3]);

m2b=2^333;
A2b=int(floor(2*(S1[3])^(2/4)))+L2;
if (floor(A2b+l2)>floor(4*(S1[3])^(2/4))):
 y22=0;
 b2=0;
 B22=0;
else:
 for i in range(l2):
  T=A2b+i;
  if (mod(int(k2*T*CC2),S2[3])<m2b):
   m2b=mod(int(k2*T*CC2),S2[3]);
   b2=T;
 y22=m2b;
 B22=mod(int(D2*b2*CC2),S2[3]);

m2c=2^333;
A2c=int(floor(2*(S1[3])^(2/4)))+L2;
```

```
if (floor(A2c+l2)>floor(4*(S1[3])^(2/4))):
 y23=0;
 b3=0;
 B23=0;
else:
 for i in range(l2):
  T=A2c+i;
  if (mod(int(k3*T*CC3),S2[3])<m2c):
   m2c=mod(int(k3*T*CC3),S2[3]);
   b3=T;
 y23=m2c;
 B23=mod(int(D3*b3*CC3),S3[3]);

y2=min(y21,y22,y23);
if ((y2==y21) and (y2!=0)):
 print "2.    y21+A2*x+B21=0 mod q"
if ((y2==y22) and (y2!=0)):
 print "2.    y22+A2*x+B22=0 mod q"
if ((y2==y23) and (y2!=0)):
 print "2.    y23+A2*x+B23=0 mod q"

m3a=2^333;
A3a=int(floor(4*(S1[3])^(3/4)))+L3;
if (floor(A3a+l3)>floor(8*(S1[3])^(3/4))):
 print "3.    L3 or l3 wrong value"
 print ""
 y31=0;
 c1=0;
 B31=0;
else:
 for i in range(l3):
  T=A3a+i;
  if (mod(int(k1*T*CC1),S1[3])<m3a):
   m3a=mod(int(k1*T*CC1),S1[3]);
   c1=T;
 y31=m3a;
 B31=mod(int(D1*c1*CC1),S1[3]);

m3b=2^333;
A3b=int(floor(4*(S1[3])^(3/4)))+L3;
if (floor(A3b+l3)>floor(8*(S1[3])^(3/4))):
```

```
 y32=0;
 c2=0;
 B32=0;
else:
 for i in range(l3):
  T=A3b+i;
  if (mod(int(k2*T*CC2),S2[3])<m3b):
   m3b=mod(int(k2*T*CC2),S2[3]);
   c2=T;
 y32=m3b;
 B32=mod(int(D2*c2*CC2),S2[3]);

m3c=2^333;
A3c=int(floor(4*(S1[3])^(3/4)))+L3;
if (floor(A3c+l3)>floor(8*(S1[3])^(3/4))):
 y33=0;
 c3=0;
 B33=0;
else:
 for i in range(l3):
  T=A3c+i;
  if (mod(int(k3*T*CC3),S2[3])<m3c):
   m3c=mod(int(k3*T*CC3),S2[3]);
   c3=T;
 y33=m3c;
 B33=mod(int(D3*c3*CC3),S3[3]);

y3=min(y31,y32,y33);
if ((y3==y31) and (y3!=0)):
 print "3.    y31+A3*x+B31=0 mod q"
if ((y3==y32) and (y3!=0)):
 print "3.    y32+A3*x+B32=0 mod q"
if ((y3==y33) and (y3!=0)):
 print "3.    y33+A3*x+B33=0 mod q"
print ""

print "q   =",S1[3]
print "x   =",A
if ((y1==y11) and (y1!=0)):
 print "y11 =",y11
if ((y1==y12) and (y1!=0)):
```

```
 print "y12 =",y12
if ((y1==y13) and (y1!=0)):
 print "y13 =",y13
if ((y2==y21) and (y2!=0)):
 print "y21 =",y21
if ((y2==y22) and (y2!=0)):
 print "y22 =",y22
if ((y2==y23) and (y2!=0)):
 print "y23 =",y23
if ((y3==y31) and (y3!=0)):
 print "y31 =",y31
if ((y3==y32) and (y3!=0)):
 print "y32 =",y32
if ((y3==y33) and (y3!=0)):
 print "y33 =",y33
if ((y1==y11) and (y1!=0)):
 print "A1  =",a1
if ((y1==y12) and (y1!=0)):
 print "A1  =",a2
if ((y1==y13) and (y1!=0)):
 print "A1  =",a3
if ((y2==y21) and (y2!=0)):
 print "A2  =",b1
if ((y2==y22) and (y2!=0)):
 print "A2  =",b2
if ((y2==y23) and (y2!=0)):
 print "A2  =",b3
if ((y3==y31) and (y3!=0)):
 print "A3  =",c1
if ((y3==y32) and (y3!=0)):
 print "A3  =",c2
if ((y3==y33) and (y3!=0)):
 print "A3 =",c3
if ((y1==y11) and (y1!=0)):
 print "B11 =",B11
if ((y1==y12) and (y1!=0)):
 print "B12 =",B12
if ((y1==y13) and (y1!=0)):
 print "B13 =",B13
if ((y2==y21) and (y2!=0)):
 print "B21 =",B21
```

```
       if ((y2==y22) and (y2!=0)):
        print "B22 =",B22
       if ((y2==y23) and (y2!=0)):
        print "B23 =",B23
       if ((y3==y31) and (y3!=0)):
        print "B31 =",B31
       if ((y3==y32) and (y3!=0)):
        print "B32 =",B32
       if ((y3==y33) and (y3!=0)):
        print "B33 =",B33
       print ""
       print "q^(3/4)/16  =",floor(((S1[3])^(3/4))/16)
       print "|v|         =",
floor((int(A)^2+int(y1)^2+int(y2)^2+int(y3)^2)^(1/2))
```

Βιβλιογραφία

[1] R. Balasubramanian and N. Koblitz: *The improbability that an elliptic curve has subexponential discrete log problem under the Menezes Okamoto Vanstone algorithm*, to appear in Journal of Cryptology.

[2] R. Barbulescu: *Algorithmes de logarithmes discrets dans les corps finis*, PhD thesis, Université de Lorraine, France, 2013.

[3] A. Becker, N. Gama and A. Joux: *Solving shortest and closest vector problems*, The decomposition approach. LMS J. Com. Math. 17 (2014), p.49-70.

[4] D. Boneh and R. Lipton: *Algorithms for blackbox fields and their applications to cryptography*, Advances in Cryptology - CRYPTO '96, Lecture Notes in Computer Science, 1109 (1992), Springer-Verlag, 283-297.

[5] I. F. Blake and T. Garefalakis: *On the security of the digital signature algorithm*, Des. Codes Cryptogr., 26 (2002), p.87-96.

[6] I. F. Blake, G. Seroussi, N. P. Smart.: *Elliptic Curve Cryptography*, Cambridge University Press, 1999.

[7] J. A. Buchmann: *Introduction to Cryptography*, New York, Berlin, Heidelberg, Springer Verlag 2001.

[8] Don Coppersmith: *Small solutions to polynomial equations, and low exponent RSA vulnerabilities.*, J. Cryptology 10 (1997), p.233-260.

[9] J. De Marrais and M. Huang: *A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields*, Algorithmic Number Theory, Lecture Notes in Computer Science, 877 (1994), Springer-Verlag, 28-40.

[10] K. Draziotis and D. Poulakis: *Lattice Attacks on DSA Schemes Based on Lagrange's Algorithm* in T. Muntean, D. Poulakis, and R. Rolland (Eds.): CAI 2013, LNCS 8080, pp. 119–131, 2013. Springer-Verlag.

[11] Konstantinos A. Draziotis: *(EC)DSA lattice attacks based on Coppersmith's method* Information Proc. Letters, Volume 116, Issue 8, Pages 541–545, Elsevier, August 2016.

[12] T. El Gamal: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, IT–31(4):469–472, July 1985.

[13] A. Enge: *Elliptic Curve and their Applications to Cryptography*, Springer, 1999.

[14] J.-L. Faugere, C. Goyet, and G. Renault: *Attacking (EC)DSA Given Only an Implicit Hint*, Selected Area of Cryptography, LNCS 7707, pp. 252-274, Springer-Verlag, Berlin - Heidelberg 2013.

[15] FIPS PUB 186-4: *Digital Signature Standard (DSS)*, July 2013.

[16] FIPS PUB 180-4: *Secure Hash Standard (SHS)*, March 2012.

[17] G. Frey and H. Ruck: *A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves*, Mathematics of Computation, 62 (1994), 865-874.

[18] W. Fulton: *Algebraic curves*, Advanced Book Classics. Addison-Wesley Publishing Company Advanced Book Program, Redwood City, CA, 1989.

[19] S. Gajbhiye, M. Sharma, S. Dashputre: *A Survey Report on Elliptic Curve Cryptography*, International Journal of Electrical and Computer Engineering (IJECE), Vol.1, No.2, pp. 195-201, December 2011.

[20] Glenn Durfee: *Cryptanalysis of RSA using algebraic and lattice methods*, Thesis, Stanford (2002).

[21] J. Hoffstein, J. Pipher and J. Silverman: *An Introduction to Mathematical Cryptography*, Springer 2008.

[22] N. Koblitz and A. Menezes: *A Survey of Public Key Cryptosystems*, Siam Review, Vol. 46, No. 4, pp. 599–634, 2004.

[23] N. Koblitz: *Good and Bad Uses of Elliptic Curve in Cryptography*, University of Washington, April 2002.

[24] Jon Kleinberg and Eva Targos: *Algorithm Design*, Person, 2006.

[25] A. Menezes, P. van Oorschot and S. Vanstene: *Handbook of Applied Cryptography*, CRC Press, 1996.

[26] A. Menezes, T. Okamoto and S. Vanstone: *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transactions on Information Theory, 39 (1993), 1639-1646.

[27] I. Mironov: *Hash functions: Theory, attacks, and applications*, Microsoft Research, November 2005.

[28] D. Micciancio and P. Voulgaris: *A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations*, Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC '10), Association for Computing Machinery, New York (2010), 351–358.

[29] V. Miller: *Uses of elliptic curves in cryptography*, Advances in Cryptology - CRYPTO '85, Lecture Notes in Computer Science, 218 (1986), Springer-Verlag, 417-426.

[30] D. Naccache, Phong Q.Nguyen, M. Tunstall and C. Whelan: *Experimenting with Faults, Lattices and the DSA*, LNCS vol. 3386 (2005), p.16-28.

[31] Nguyen, P.Q., Shparlinski, I.E: *The Insecurity of the Digital Signature Algorithm with Partially Known Nonces.*, J. Cryptology 15, 151–176 (2002).

[32] Nicolas Gama and Phong Q. Nguyen: *Predicting Lattice Reduction*, Eurocrypt 2008, LNCS vol. 4965 (2008), p.31-51.

[33] P. van Oorschot and M. Wiener: *Parallel collision search with cryptanalytic applications*, to appear in Journal of Cryptology.

[34] S. Pohlig and M. Hellman: *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Transactions on Information Theory, 24 (1978), 106-110.

[35] J. Pollard: *J. Pollard, Monte Carlo methods for index computation mod p*, Mathematics of Computation, 32 (1978), 918-924.

[36] D. Poulakis: *Some Lattice Attacks on DSA and ECDSA Applicable Algebra in Engineering* Communication and Computing 22, 347–358 (2011).

[37] D. Poulakis: *New lattice attacks on DSA schemes* Journal of Mathematical Cryptology, Vol 10, Issue 2, Jun 2016, Pages 135–144.

[38] C. Ritzenthaler: *Elliptic curves and applications to cryptography*, University at Marseille, 1st Semester 2011-2012.

[39] K. Rosen: *Elliptic Curves Number Theory and Cryptography*, Washington, 2008.

[40] H. Ruck: *On the discrete logarithm in the divisor class group of curves*, preprint, 1997.

[41] C. F. Shannon: *Communication theory of secrecy systems*, Bell Systems Technical Journal, 28(1949), pp. 656-715.

[42] T. Satoh and K. Araki: *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*, preprint, 1997.

[43] C. L. Siegel: *A mean value theorem in geometry of numbers*, Ann. of Math. (2), 46:340–347, 1945.

[44] M. Sipser: *Introduction to the Theory of Computation*, PWS Publishing Co., Boston, Massachusetts, 1997.

[45] I. Semaev: *Evaluation of discrete logarithms on some elliptic curves*, to appear in Mathematics of Computation.

[46] N. Smart: *The discrete logarithm problem on elliptic curves of trace one*,Journal of Cryptology, 12 (1999), pp. 193-196.

[47] A. Stein: *Equivalences between elliptic curves and real quadratic congruence function fields*, presented at Pragocrypt '96.

[48] W. A. Stein et al: *Sage Mathematics Software*,The Sage Development Team.

[49] Α. Φυραρίδη Κ. Μέξη: *Αλγεβρικές Δομές Ι*, Ελληνικά Ακαδημαικά Συγγράμματα, Ιωάννινα, 1990.

[50] Δ. Πουλάκη: *Κρυπτογραφία*, Εκδόσεις Ζήτη, Θεσσαλονίκη, 2004.

[51] Δ. Πουλάκη: *Υπολογιστική Θεωρία Αριθμών*, Ελληνικά Ακαδημαικά Ηλεκτρονικά Συγγράμματα και Βοηθήματα, 2015.

[52] *https://en.wikipedia.org/wiki/Digital_signature*

[53] *http://www.sagemath.org/*