ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

$\mu\Pi\lambda\forall$-Μεταπτυχιακο Προγραμμα Λογικης & Αλγοριθμων

Παραγοντοποίηση Ακεραίων με Ελλειπτικές Καμπύλες

Κατερίνα Μ. Κούτα

Επιβλέπων Καθηγητής Ευάγγελος Ράπτης

Διπλωματικη Εργασια

Ακαδημαϊκό Έτος 2008 - 2009

Περιεχόμενα

Κεφάλαιο 1

Πρόλογος

Η παρούσα εργασία στοχεύει στην περιγραφή και ανάλυση ενός αλγόριθμου παραγοντοποίησης θετικών ακέραιων αριθμών. Ο αλγόριθμος αυτός ονομάζεται *αλγόριθμος παραγοντοποίησης του Lenstra ή μέθοδος παραγοντοποίησης με ελλειπτικές καμπύλες (Elliptic Curve Method - ECM) και αναπτύχθηκε από τον Hendrik W. Lenstra στα 1987. Πρόκειται για έναν γρήγορο υποεκθετικού χρόνου αλγόριθμο, ο οποίος εξαρτάται από τη χρήση ελλειπτικών καμπύλων.*

Στην πορεία φάνηκε ότι η ECM πρακτικά είναι αλγόριθμος παραγοντοποίησης με ειδική στόχευση καθώς είναι ιδιαίτερα αποτελεσματικός για την παραγοντοποίηση αριθμών με περί τα 60 ψηφία. Για παραγοντοποίηση τυχαίου αριθμού, η ECM είναι η τρίτη πιο γρήγορη γνωστή μέθοδος. Η δεύτερη πιο γρήγορη είναι η μέθοδος του τετραγωνικού κόσκινου (quadratic sieve) και η ταχύτερη όλων είναι ο αλγόριθμος του γενικού αριθμητικού κόσκινου (general number field sieve).

Για αριθμούς με περισσότερα από 60 ψηφία, ο αλγόριθμος του Lenstra αποτελεί επί του παρόντος τον βέλτιστο γνωστό αλγόριθμο για την εύρεση μικρών διαιρετών τους, οι οποίοι έχουν από 20 εώς και 30 ψηφία (περί τα 64 με 83 bits περίπου), καθώς ο χρόνος εκτέλεσής του εξαρτάται περισσότερο από το μέγεθος του μικρότερου παράγοντα p παρά από το μέγεθος του αριθμού n που πρέπει να παραγοντοποιηθεί. Οπότε, συχνά η μέθοδος ελλειπτικών καμπύλων του Lenstra χρησιμοποιείται για την απομόνωση μικρών παραγόντων από έναν ιδιαίτερα μεγάλο ακέραιο με πολλούς παράγοντες και αν ο ακέραιος που υπολοίπεται είναι σύνθετος, τότε θα έχει μόνο μεγάλους παράγοντες και παραγοντοποιείται περεταίρω με χρήση άλλων μεθόδων.

Ο μεγαλύτερος παράγοντας που έχει βρεθεί ως σήμερα με τη χρήση του ECM αλγόριθμου ανακαλύφθηκε στις 24 Αυγούστου 2006 από τον B. Dodson είναι παράγοντας του $10^{381} + 1$ και έχει 67 ψηφία [1]*. Η αύξηση του πλήθους των καμπύ-*

[1] Πρόκειται για τον αριθμό 44443497921567099078957525517986319089461806087687379462802380788881

λων που ελέγχονται παράλληλα, βελτιώνει την πιθανότητα εύρεσης ενός παράγοντα, αλλά όχι γραμμικά ως προς την αύξηση του πλήθους των ψηφίων.

Η μέθοδος του Lenstra βασίστηκε στην $(p-1)$ μέθοδο του Pollard (1974), αντικαθιστώντας όμως την πολλαπλασιαστική ομάδα με την ομάδα των σημείων μίας τυχαίας ελλειπτικής καμπύλης. Στη χειρότερη περίπτωση, δηλαδή όταν ο n είναι γινόμενο δύο πρώτων ίδιας τάξης μεγέθους, ο αναμενόμενος χρόνος για την εκτέλεση του αλγόριθμου του H. W. Lenstra είναι της τάξης του $\exp\left((1+\mathcal{O}(1))\sqrt{\log n \log\log n}\right)$ (για $n \to \infty$). Υπάρχουν διάφοροι άλλοι αλγόριθμοι παραγοντοποίησης θετικών ακέραιων, των οποίων ο αναμενόμενος χρόνος εκτέλεσης έχει τον ίδιο τύπο. Ωστόσο, για αυτούς τους αλγόριθμους ο χρόνος εκτέλεσης είναι ανεξάρτητος από το μέγεθος των πρώτων παραγόντων του n, ενώ η μέθοδος των ελλειπτικών καμπύλων είναι ουσιαστικά ταχύτερη για μικρό πρώτο παράγοντα p.

Μετά την ολοκλήρωση της μελέτη μας, παραθέτουμε στο Παράρτημα, μία παρουσίαση του αλγόριθμου τετραγωνικού κόσκινου (quadratic sieve algorithm), ο οποίος αποδίδει καλύτερα από τη μέθοδο των ελλειπτικών καμπύλων για περιπτώσεις όπου οι δύο πρώτοι παράγοντες του προς παραγοντοποίηση αριθμού είναι της ίδιας τάξης μεγέθους.

Κλείνοντας αυτό τον πρόλογο, θα ήθελα να απευθύνω ευχαριστίες στους καθηγητές μου στο Μ.Π.Λ.Α., χάρη στη διδασκαλία των οποίων πήρα τις βάσεις αλλά και τα ερεθίσματα για να κάνω μια εισαγωγή σε πεδία ιδιαίτερα ενδιαφέροντα, την ύπαρξη των οποίων αγνοούσα. Κυρίως, θα ήθελα να ευχαριστήσω τον κύριο Βαγγέλη Ράπτη, Καθηγητή του Τμήματος Μαθηματικών του Πανεπιστημίου Αθηνών, επιβλέποντα καθηγητή της εργασίας αυτής, για τη συνεργασία που είχαμε κατά την εκπόνησή της και για το γεγονός ότι δέχτηκε να αναλάβει αυτό τον ρόλο σε στενά χρονικά πλαίσια. Τέλος, αφιερώνω την εργασία αυτή στους γονείς μου, Αθανασία και Μιχάλη και στον αδελφό μου, Γιώργο.

Αθήνα, Μάης 2009.

Κατερίνα Μ. Κούτα.

Κεφάλαιο 2

Ελλειπτικές Καμπύλες

Στα μέσα της δεκαετίας του 1980, οι Miller και Koblitz εισήγαγαν τις ελλειπτικές καμπύλες στην κρυπτογραφία, και ο Lenstra έδειξε πώς μπορεί κανείς να χρησιμοποιήσει τις ελλειπτικές καμπύλες για να παραγοντοποιήσει ακέραιους.

2.1 Εισαγωγικά

Ορισμός 2.1.1 *Μία ελλειπτική καμπύλη $\mathbb{E}$ πάνω από ένα σύνολο $\mathbf{K}$ είναι το γράφημα μίας εξίσωσης της μορφής*

$$\mathbb{E} : y^2 = x^3 + ax + b\,, \tag{2.1}$$

όπου a και b είναι σταθερές που ανήκουν στο σύνολο $\mathbf{K}$. Η (2.1) θα αναφέρεται ως η **εξίσωση Weierstrass** *μίας ελλειπτικής καμπύλης. Επιπλέον, για τεχνικούς λόγους, είναι χρήσιμο να συμπεριλάβουμε και ένα* **«σημείο στο άπειρο»** *σε κάθε ελλειπτική καμπύλη.*

Σημείωση 2.1.2 *Αξίζει να επισημανθεί ότι ο όρος «ελλειπτική καμπύλη» είναι κατά μία έννοια παραπλανητικός. Οι ελλειπτικές καμπύλες δεν είναι ελλείψεις. Πήραν την ονομασία τους από το συσχετισμό τους με τα ελλειπτικά ολοκληρώματα, όπως είναι τα*

$$\int_{z_1}^{z_2} \frac{dx}{\sqrt{x^3+ax+b}} \quad \text{και} \quad \int_{z_1}^{z_2} \frac{x\,dx}{\sqrt{x^3+ax+b}}\,,$$

τα οποία ανακύπτουν κατά τον υπολογισμό του μήκους τόξου των ελλείψεων, και τα οποία οδήγησαν στις ελλειπτικές καμπύλες.

2.2 Σημείο στο Άπειρο

Το σημείο στο άπειρο είναι πιο εύκολο να το θεωρήσουμε ως ένα σημείο (∞, ∞), *το οποίο συνήθως συμβολίζουμε απλά ως* ∞ *και το φανταζόμαστε συνήθως να κάθεται στην κορυφή του άξονα* yy'. *Για υπολογιστικούς σκοπούς, το σημείο στο άπειρο θα είναι ένα σύμβολο που θα ικανοποιεί ορισμένους υπολογιστικούς κανόνες. Επί παραδείγματι, μία ευθεία θα λέμε ότι διέρχεται από το σημείο στο άπειρο εάν και μόνο εάν η ευθεία αυτή είναι κατακόρυφη (δηλαδή όταν* $x =$ *σταθερά). Είναι αναμενόμενο το σημείο* ∞ *να φαίνεται ελαφρώς αφύσικο, θα δούμε όμως στη συνέχεια, ότι η συμπερίληψή του στα σημεία κάθε ελλειπτικής καμπύλης έχει χρήσιμες συνέπειες.*

Σημειώνεται εδώ ότι, εν προκειμένω, δεχόμαστε επιπλέον την σύμβαση ότι το κατώτερο άκρο του yy' *ταυτίζεται με το ανώτερο άκρο, οπότε το* ∞ *κάθεται και στο κατώτερο άκρο του* yy'. *Με άλλα λόγια, φανταζόμαστε τα άκρα του άξονα* yy' *να τυλίγονται και να συναντιούνται(ίσως κάπου στο πίσω μέρος της σελίδας) στο σημείο* ∞. *Αυτό αναμένεται να φαίνεται περίεργο. Ωστόσο, αν δουλεύουμε σε κάποιο σώμα, άλλο από το σώμα* $\mathbb{R}$ *των πραγματικών αριθμών, λόγου χάρη ένα πεπερασμένο σώμα, μάλλον δεν θα υπάρχει έννοια διάταξης των στοιχείων και, κατ' επέκταση, δεν θα έχει νόημα η διάκριση του ανώτερου άκρου του* yy' *από το κατώτερο.*

Στην πράξη, σε αυτή την περίπτωση, τα άκρα του yy' *δεν έχουν μαθηματική υπόσταση, παρεκτός αν εισάγουμε προβολική γεωμετρία, καθώς το σημείο στο άπειρο μπορεί να μελετηθεί σχολαστικά στα πλαίσια της προβολικής γεωμετρίας, ώστε να γίνει πιο σαφής και αυστηρός ο ορισμός του.*

Η προβολική γεωμετρία μελετά τις ιδιότητες γεωμετρικών αντικειμένων που παραμένουν αμετάβλητες υπό προβολή. Για παράδειγμα, προβολικός 2-χώρος πάνω από ένα σώμα F, *ο οποίος συμβολίζεται με* $\mathbb{P}^2(F)$, *είναι το σύνολο* $\{(x, y, z) : x, y, z \in F\} - \{(0, 0, 0)\}$ *όλων των κλάσεων ισοδυναμίας των προβολικών σημείων* $(tx, ty, tz) \sim (x, y, z)$ *για μη μηδενικό* $t \in F$. *Οπότε, αν* $z \neq 0$, *τότε υπάρχει ένα μοναδικό προβολικό σημείο στην κλάση του* (x, y, z) *της μορφής* $(x, y, 1)$, *δηλαδή το* $(x/z, y/z, 1)$. *Επομένως, το* $\mathbb{P}^2(F)$ *μπορεί να ταυτιστεί με όλα τα σημεία του συνηθισμένου ή του affine επιπέδου μαζί με τα σημεία για τα οποία* $z = 0$. *Τα τελευταία είναι τα σημεία της ευθείας στο άπειρο, την οποία μπορεί κανείς να την φανταστεί ως τον ορίζοντα του επιπέδου. Με αυτόν τον ορισμό, μπορεί κανείς να δει ότι το σημείο στο άπειρο στον Ορισμό 2.1.1 είναι το* $(0, 1, 0)$ *στο* $\mathbb{P}^2(F)$. *Πρόκειται για την τομή του άξονα* y *με την ευθεία στο άπειρο.*

Στα πλαίσια, ωστόσο, της παρούσας εργασίας ο διαισθητικός ορισμός του κρίνεται ως επαρκής.

Τέλος, ορίζουμε ότι δύο κατακόρυφες ευθείες τέμνονται στο ∞. Λόγω συμμετρίας, αν τέμνονται στο ανώτερο άκρο του yy', τότε θα τέμνονται και στο κατώτερο άκρο. Όμως δύο διακεκριμένες ευθείες μπορούν να τέμνονται σε το πολύ ένα σημείο, άρα το «άνω» σημείο ∞ με το «κάτω» σημείο ∞ θα πρέπει να ταυτίζονται. Αυτή είναι μία χρήσιμη, για την εργασία μας, ιδιότητα του σημείου στο άπειρο.

2.3 Ορισμοί, Πράξεις & Βασικές Προτάσεις

Ένα βασικό σημείο είναι ο προσδιορισμός του σύνολου $\mathbf{K}$ όπου παίρνουν τιμές τα a, b, x και y. Συνήθως, θεωρούμε $\mathbf{K}$ να είναι ένα σώμα, για παράδειγμα, το σύνολο των πραγματικών αριθμών $\mathbb{R}$, το σύνολο των μιγαδικών αριθμών $\mathbb{C}$, το συνόλο των ρητών αριθμών $\mathbb{Q}$, ένα από τα πεπερασμένα σώματα $\mathbb{F}_p$ $(\mathbb{Z}_p)$ για κάποιον πρώτο p, ή ένα από τα πεπερασμένα σώματα $\mathbb{F}_q$, όπου $q = p^k$ με $k > 1$.

Αν $\mathbf{K}$ είναι σώμα τέτοιο ώστε $a, b \in \mathbf{K}$, τότε λέμε ότι η ελλειπτική καμπύλη $\mathbb{E}$ ορίζεται πάνω από το $\mathbf{K}$. Σε περίπτωση όπου επιθυμούμε να αναφερθούμε σε σημεία με συντεταγμένες σε κάποιο σώμα $\mathbb{L}$ με $\mathbb{L} \supseteq \mathbf{K}$, γράφουμε $\mathbb{E}(\mathbb{L})$. Εξ ορισμού, αυτό το σύνολο πάντα περιέχει το σημείο ∞, οπότε είναι

$$\mathbb{E}(\mathbb{L}) \;=\; \{\infty\} \cup \{(x, y) \in \mathbb{L} \times \mathbb{L} \,|\, y^2 = x^3 + ax + b\}\,.$$

Αν, για παράδειγμα, δουλεύουμε πάνω από τους πραγματικούς αριθμούς, το γράφημα $\mathbb{E}$ έχει δύο πιθανές μορφές, ανάλογα με το αν το κυβικό πολυώνυμο ως προς x έχει μία πραγματική ρίζα ή τρεις πραγματικές ρίζες.

Εν γένει, επιλέγουμε το κυβικό πολυώνυμο στο δεξί μέλός της εξίσωσης (2.1.1) έτσι ώστε να μην έχει ρίζες πολλαπλότητας μεγαλύτερης της μονάδας. Αυτό σημαίνει ότι αποκλείουμε, για παράδειγμα το γράφημα της εξίσωσης $y^2 = x^2(x-1)$. Στην πράξη, θέτουμε τον περιορισμό

$$4a^3 + 27b^2 \neq 0$$

και μπορεί να αποδειχθεί ότι αν οι ρίζες του κυβικού πολυωνύμου είναι r_1, r_2, r_3, τότε η διακρίνουσά του είναι

$$((r_1 - r_2)(r_1 - r_3)(r_2 - r_3))^2 \;=\; -(4a^3 + 27b^2)\,,$$

οπότε οι ρίζες του κυβικού πολυωνύμου πρέπει να είναι διακεκριμένες.

Η αντίθετη περίπτωση βέβαια, παρουσιάζει επίσης κάποιο ενδιαφέρον στην παραγοντοποίση ακεραίου $n = pq$. Αν, δηλαδή, δουλεύουμε $\bmod n$, η εξίσωση της

$\mathbb{E}$ έχει πολλαπλή ρίζα $\bmod n$ εάν η διακρίνουσά της είναι ισοϋπόλοιπη με $0 \bmod n$. Εφόσον ο n είναι σύνθετος, υπάρχει και η ενδιάμεση περίπτωση, όπου ο μέγιστος κοινός διαιρέτης του n και της διακρίνουσας δεν είναι ούτε 1 ούτε n. Αυτό όμως μας δίνει έναν μη τετριμμένο πραράγοντα του n, οπότε, σε αυτή την περίπτωση, δεν χρειάζεται να υλοποιηθεί ο αλγόριθμος παραγοντοποίης.

Εξίσωση Ελλειπτικής Καμπύλης

Για να έχουμε περισσότερη ευελιξία, μπορούμε να επιτρέψουμε και εξισώσεις κάπως πιο γενικές από την εξίσωση Weierstrass που είδαμε στον ορισμό 2.1.1, της μορφής

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5\,, \tag{2.2}$$

όπου τα $a_1, \cdots, a_5$ είναι σταθερές. Αυτή η πιο γενική μορφή που δίνεται μέσω της (2.2) ονομάζεται **γενικευμένη εξίσωση Weierstrass** *και είναι χρήσιμη όταν δουλεύουμε με σώματα χαρακτηριστικής 2 και χαρακτηριστικής 3.*

Ειδικά, εάν η χαρακτηριστική του σώματος δεν είναι 2, διαιρώντας με 2 και συμπληρώνοντας τετράγωνα παίρνουμε

$$\left(y + \frac{a_1x}{2} + \frac{a_3}{2}\right)^2 = \left(a_2 + \frac{a_1^2}{4}\right)x^2 + \left(a_4 + \frac{a_1a_3}{2}\right)x + \left(\frac{a_3^2}{4} + a_5\right)\,, \tag{2.3}$$

το οποίο, εφόσον δουλεύουμε σε σώμα, θέτοντας $y_1 = y + a_1x/2 + a_3/2$ και χρησιμοποιώντας τις σταθερές a_2', a_4', a_6', μπορεί να γραφθεί ως

$$y_1^2 = x^3 + a_2'x^2 + a_4'x + a_6'\,,$$

Εάν, επιπλέον, η χαρακτηριστική δεν είναι 3, τότε μπορούμε να θέσουμε $x = x_1 - a_2'/3$, οπότε

$$\begin{aligned}
y_1^2 &= \left(x_1 - \frac{a_2'}{3}\right)^3 + a_2'\left(x_1 - \frac{a_2'}{3}\right)^2 + a_4'\left(x_1 - \frac{a_2'}{3}\right) + a_6' \\
&= \left(x_1^3 - 3x_1^2\frac{a_2'}{3} + 3x_1\frac{a_2'^2}{3^2} - \frac{a_2'^3}{3^3}\right) + a_2'\left(x_1^2 - 2x_1\frac{a_2'}{3} + \frac{a_2'^2}{3^2}\right) + a_4'\left(x_1 - \frac{a_2'}{3}\right) + a_6' \\
&= x_1^3 + 3x_1\frac{a_2'^2}{3^2} - \frac{a_2'^3}{3^3} - a_2'2x_1\frac{a_2'}{3} + a_2'\frac{a_2'^2}{3^2} + a_4'x_1 - a_4'\frac{a_2'}{3} + a_6' \\
&= x_1^3 + \left(\frac{a_2'^2}{3} - \frac{2a_2'^2}{3} + a_4'\right)x_1 + \left(a_6' - \frac{a_2'a_4'}{3} - \frac{a_2'^3}{3^3} + \frac{a_2'^3}{3^2}\right)
\end{aligned}$$

δηλαδή καταλήγουμε στην εξίσωση Weierstrass (2.1), για κατάλληλες σταθερές a και b.

Στην παρούσα εργασία δε θα μας απασχολήσουν εξισώσεις της μορφής (2.2) ή (2.3), καθώς θα δουλέψουμε με πεπερασμένα σώματα $\mathbb{F}_q$, όπου $q = p^r$, για κάποιον πρώτο $p \geq 5$. Οπότε, για τη μελέτη μας, αρκεί η πιο απλή και εύχρηστη μορφή εξίσωσης (2.1). □

Ας μελετήσουμε τώρα κάποια βασικά σημεία για το σύνολο των σημείων μίας ελλειπτικής καμπύλης.

Πρόταση 2.3.1 *Σε τυχαία ελλειπτική καμπύλη , ξεκινώντας με δύο σημεία της, ή ακόμα και ένα, μπορούμε να παράγουμε ένα νέο σημείο της καμπύλης αυτής.*

Απόδειξη. *Έστω ελλειπτική καμπύλη $\mathbb{E}$ που δίνεται από την εξίσωση $y^2 = x^3 + ax + b$ και δύο σημεία $P_1 = (x_1, y_1)$ και $P_2 = (x_2, y_2)$ πάνω σε αυτήν. Ορίζουμε ένα νέο σημείο P_3 της $\mathbb{E}$ ως εξής. Θεωρούμε την ευθεία L που διέρχεται από τα P_1 και P_2. Θα δείξουμε ότι η L τέμνει την $\mathbb{E}$ σε ένα τρίτο σημείο P_3'. Ανακλούμε το P_3' ως προς τον άξονα $x'x$ (δηλαδή, αλλάζουμε πρόσημο στην τεταγμένη του) και παίρνουμε το σημείο P_3. Ορίζουμε τη διμελή πράξη $+_{\mathbb{E}}$ ως ακολούθως:*

$$P_1 +_{\mathbb{E}} P_2 \;=\; P_3\,.$$

Παρατηρούμε αμέσως ότι η πράξη $+_{\mathbb{E}}$ δεν ταυτίζεται με την πρόσθεση σημείων κατά συντεταγμένες. Στα ακόλουθα, προς απλοποίηση των συμβολισμών, αντί του $+_{\mathbb{E}}$ θα χρησιμοποιούμε το σύμβολο $+$.

Ας υποθέσουμε, κατ' αρχήν, ότι $P_1 \neq P_2$ και κανένα εκ των P_1, P_2 δεν είναι το σημείο ∞. Θεωρούμε την ευθεία L που περνάει από τα P_1 και P_2. Η κλίση της L δίνεται ως

$$m \;=\; \frac{y_2 - y_1}{x_2 - x_1}\,.$$

Αν $x_1 = x_2$, τότε η L είναι οριζόντια. Θα εξετάσουμε την περίπτωση αυτή στη συνέχεια. Οπότε, ας υποθέσουμε ότι $x_1 \neq x_2$, άραη εξίσωση της L είναι

$$y \;=\; m(x - x_1) + y_1\,.$$

Για να βρούμε το σημείο τομής της L με την $\mathbb{E}$ αντικαθιστούμε το y από την τελευταία εξίσωση στην εξίσωση της ελλειπτικής καμπύλης και έχουμε

$$(m(x - x_1) + y_1) \;=\; x^3 + ax^2 + b\,,$$

το οποίο μπορεί να γραφθεί στη μορφή

$$0 \;=\; x^3 - m^2x^2 + (a + 2m^2x_1 - 2my_1)x + (b - m^2x_1 + 2mx_1y_1 - y_1^2)\,, \tag{2.4}$$

όπου οι τρεις ρίζες του κυβικού πολυωνύμου στο δεξί μέλος της παραπάνω εξίσωσης αντιστοιχούν στα σημεία τομής της L με την $\mathbb{E}$. Γενικά, η επίλυση κυβικού πολυωνύμου δεν είναι τετριμμένη υπόθεση, αλλά στην τρέχουσα περίπτωση γνωρίζουμε ήδη τις δύο ρίζες, δηλαδή τα x_1 και x_2, εφόσον τα P_1 και P_2 είναι κοινά σημεία για τις L και $\mathbb{E}$. Επομένως, θα μπορούσαμε να παραγοντοποιήσουμε το κυβικό πολυώνυμο για να πάρουμε την τρίτη τιμή ως προς x, έστω x'_3. Θα μπορούσαμε δηλαδή να γράψουμε το κυβικό ως $(x - x_1)(x - x_2)(x - x'_3)$. Αλλά υπάρχει και απλούστερος τρόπος, καθώς γνωρίζουμε ότι ισχύει

$$x_1 + x_2 + x'_3 \;=\; -(-m^2).$$

Απ' όπου παίρνουμε

$$x'_3 \;=\; m^2 - x_1 - x_2\,.$$

Επειδή το $P'_3 = (x'_3, y'_3)$ είναι σημείο της L ισχύει και ότι $y'_3 = m(x - x_1) + y_1$. Οπότε, ανακλώντας το P'_3 ως προς τον άξονα $x'x$ παίρνουμε το σημείο $P_3 = (x_3, y_3)$:

$$x_3 \;=\; m^2 - x_1 - x_2 \quad \textit{και} \quad y_3 \;=\; m(x_1 - x_3) - y_1\,.$$

Στην περίπτωση όπου $x_1 = x_2$ αλλά $y_1 \neq y_2$, η ευθεία που συνδέεται τα P_1 και P_2 είναι μία κατακόρυφη ευθεία, άρα η ευθεία αυτή τέμνει την $\mathbb{E}$ στο ∞. Η ανάκλαση του ∞ ως προς τον άξονα $x'x$ δίνει το ίδιο το ∞ (σύμφωνα με τη συνθήκη που ισχύει για το ∞). Άρα σε αυτή την περίπτωση, $P_1 + P_2 = \infty$.

Εξετάζουμε τώρα την περίπτωση όπου $P_1 = P_2 = (x_1, y_1)$. Όταν δύο σημεία σε μία καμπύλη είναι πολύ κοντά το ένα στο άλλο, η ευθεία που διέρχεται από αυτά προσεγγίζει μία εφαπτόμενη στην καμπύλη ευθεία. Συνεπώς, όταν τα δύο σημεία ταυτίζονται, η ευθεία L που τα συνδέει θεωρούμε ότι είναι εφαπτομένη της καμπύλης. Αν $y_1 = 0$, τότε η ευθεία L είναι κατακόρυφη και θέτουμε $P_1 + P_2 = \infty$, ακριβώς με το ίδιο επιχείρημα που χρησιμοποιήσαμε στην περίπτωση $x_1 = x_2$. Αλλιώς, αν $y_1 \neq 0$, παραγωγίζουμε την εξίσωση της ελλειπτικής καμπύλης ως προς x και υπολογίζουμε τον συντελεστή διεύθυνσης m της L.

$$2y\frac{dy}{dx} \;=\; 3x^2 + a\,, \quad \textit{άρα} \quad m \;=\; \frac{dy}{dx} \;=\; \frac{3x_1^2 + a}{2y_1}\,.$$

Η εξίσωση της L είναι $y = m(x - x_1) + y_1$, όπως και στην πρώτη περίπτωση που διακρίναμε. Με ανάλογα βήματα, παίρνουμε την εξίσωση (2.4). Αυτή τη φορά γνωρίζουμε μία ρίζα του κυβικού πολυωνύμου στο δεξί μέλος της (2.4), δηλαδή την x_1, αλλά πρόκειται για διπλή ρίζα, αφού η L είναι εφαπτομένη της $\mathbb{E}$ στο P_1. Επομένως, βρίσκουμε το σημείο $P_3 = (x_3, y_3) = P_1 + P_1$ ως εξής

$$x_3 \;=\; m^2 - 2x_1 \quad \textit{και} \quad y_3 \;=\; m(x_1 - x_3) - y_1\,.$$

Τελευταία θεωρούμε την περίπτωση κατά την οποία $P_2 = \infty$. Η ευθεία που διέρχεται από τα P_1 και ∞ είναι μία κατακόρυφη ευθεία, η οποία τέμνει την $\mathbb{E}$ στο σημείο P_1', το οποίο είναι η ανάκλαση του P_1 ως προς τον άξονα $x'x$. Με την ανάκλαση του P_1' ως προς τον $x'x$ για να πάρουμε το $P_3 = P_1 + \infty$, επιστρέφουμε στο P_1. Συνεπώς

$$P_1 + \infty \;=\; P_1$$

για κάθε σημείο P_1 που ανήκει στην $\mathbb{E}$. Φυσικά, επεκτείνουμε αυτό το συμπέρασμα ώστε να συμπεριλαμβάνει και τη διαπίστωση $\infty + \infty = \infty$, καθώς ο συλλογισμός μας αφορούσε τυχαίο σημείο P_1, χωρίς να αποκλείει το $P_1 = \infty$. □

Συνοψίζοντας, από τα παραπάνω παίρνουμε το εξής Λήμμα, αναφορικά με τον ορισμό της πρόσθεσης σημείων πάνω σε ελλειπτική καμπύλη.

Λήμμα 2.3.2 *Έστω $\mathbb{E}$ μία ελλειπτική καμπύλη που ορίζεται από την εξίσωση $y^2 = x^3 + ax + b$. Ας είναι $P_1 = (x_1, y_1)$ και $P_2 = (x_2, y_2)$ σημεία της $\mathbb{E}$ με $P_1 \neq \infty \neq P_2$. Ορίζουμε την πράξη $P_1 + P_2 = P_3 = (x_3, y_3)$ ως εξής:*

1. *Αν $x_1 \neq x_2$, τότε*

$$x_3 \;=\; m^2 - x_1 - x_2, \qquad y_3 \;=\; m(x_1 - x_3) - y_1 \;\; \text{όπου} \;\; m \;=\; \frac{y_2 - y_1}{x_2 - x_1}\,.$$

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

3. *Αν $P_1 = P_2$ και $y_1 \neq 0$, τότε*

$$x_3 \;=\; m^2 - 2x_1, \qquad y_3 \;=\; m(x_1 - x_3) - y_1 \;\; \text{όπου} \;\; m \;=\; \frac{3x_1^2 + a}{2y_1}\,.$$

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

5. *Επιπλέον, ορίζουμε για κάθε P στην $\mathbb{E}$*

$$P \;+\; \infty \;=\; P\,.$$

□

Παρατήρηση 2.3.3 *Όταν οι συντεταγμένες των P_1 και P_2 ανήκουν σε ένα σώμα $\mathbf{K}$ που περιέχει τις σταθερές a και b, τότε το σημείο $P_1 + P_2$ έχει επίσης συντεταγμένες από το $\mathbf{K}$. Οπότε η $\mathbb{E}(\mathbf{K})$ είναι κλειστή ως προς την πρόσθεση σημείων την οποία μόλις ορίσαμε.*
Ειδικότερα, όταν το $\mathbf{K}$ είναι κάποιο πεπερασμένο σώμα, καθώς τα σημεία της $\mathbb{E}(\mathbf{K})$ σχηματίζουν υποσύνολο του $\mathbf{K}$, έπεται ότι το $\mathbb{E}(\mathbf{K})$ είναι πεπερασμένο σύνολο.

Θεώρημα 2.3.4 *Για κάθε ελλειπτική καμπύλη $\mathbb{E}$ πάνω από σώμα $\mathbf{K}$, η οποία ορίζεται μέσω της εξίσωσης Weierstrass (2.1), η αλγεβρική δομή $< \mathbb{E}(\mathbf{K}), +_{\mathbb{E}} >$ συνιστά προσθετική αβελιανή ομάδα, όπου το σημείο στο άπειρο ∞ είναι το ταυτοτικό στοιχείο της ομάδας.*

Απόδειξη. *Η αντιμεταθετικότητα είναι προφανής, τόσο από τους τύπους υπολογισμού του αθροίσματος όσο και υπό την έννοια ότι η ευθεία που διέρχεται από δύο σημεία P_1 και P_2 είναι η ίδια με την ευθεία που διέρχεται από τα σημεία P_2 και P_1. Ο ορσιμός του σημείου στο άπειρο ως ταυτοτικού στοιχείου ισχύει εξ ορισμού του ∞. Ως προς την ιδιότητα ύπαρξης αντίθετου στοιχείου P' για κάθε στοιχείο P της $\mathbb{E}$, παρατηρούμε ότι αν για κάθε P στην $\mathbb{E}$ θεωρήσουμε την ανάκλαση P' του P ως προς τον άξονα $x'x$, τότε, όπως είδαμε, ισχύει $P + P' = \infty$.*
Αρκεί να δείξουμε ότι ισχύει η προσεταιριστική ιδιότητα, δηλαδή ότι για τυχαία σημεία P_1, P_2, P_3 στην $\mathbb{E}$ ισχύει ότι

$$(P_1 + P_2) + P_3 \;=\; P_1 + (P_2 + P_3)\,.$$

Η ιδιότητα της προσεταιριστικότητας μπορεί να αποδειχθεί μέσω υπολογισμών από τους τύπους, αλλά η μέθοδος αυτή έχει το μειονέκτημα ότι χρειάζεται να διακρίνουμε αρκετές περιπτώσεις, πχ ανάλογα με τον αν ισχύει η όχι $P_1 = P_2$, ή $P_3 = (P_1 + P_2)$, κόκ. Έτσι ενώ η μέθοδοες είναι αποτελεσματική, δεν είναι ιδιαίτερα κομψη. Μια άλλη απόδειξη, πιο τεχνική και συνάμα πιο κομψή, μπορεί να δοθεί με προσέγγιση μέσω προβολικής γεωμετρίας, αλλά αυτό ξεφεύγει από τον στόχο της παρούσας εργασίας. Εν προκειμένω, στα ακόλουθα, θα δεχθούμε ότι η προσεταιριστική ιδιότητα ισχύει. □

Παρατήρηση 2.3.5 *Εκτός από πρόσθεση σημείων, μπορούμε να κάνουμε και αφαίρεση σημείων πάνω από μια ελλειπτική καμπύλη $\mathbb{E}$. Αρκεί να παρατηρήσει κανείς ότι η ευθεία που διέρχεται από τα σημεία (x, y) και $(x, -y)$ είναι κατακόρυφη, οπότε το τρίτο σημείο τομής με την $\mathbb{E}$ είναι το σημείο στο άπειρο. Η ανάκλαση ως προς τον άξονα $x'x$ δίνει και πάλι το ∞. Επομένως, είναι*

$$(x,\, y) \,+\, (x,\, -y) \;=\; \infty\,.$$

Εφόσον το ∞ είναι το ουδέτερο στοιχείο της ομάδας $< \mathbb{E}, + >$, ορίζουμε

$$-\,(x,\, y) \;=\; (x,\, -y)\,.$$

Με αυτόν τον ορισμό, για να βρούμε το εξαγόμενο της διαφοράς $P - Q$ για δύο σημεία $P, Q \in \mathbb{E}$, αρκεί να υπολογίσουμε το άθροισμα $P + (-Q)$. □

2.4 Πλήθος Σημείων mod**n**

Έστω $\mathbb{E} : y^2 \equiv x^3 + ax + b \pmod{q}$ *μια ελλειπτική καμπύλη, όπου* $q = p^r$, r *θετικός ακέραιος και* p *πρώτος με* $p \geq 5$. *Είναι αρκετά απλό να δει κανείς ότι στο σώμα* $\mathbb{F}_q$, *η καμπύλη* $\mathbb{E}$ *μπορεί να έχει πλήθος σημείων το πολύ* $2q+1$, *δηλαδή το σημείο στο άπειρο και* $2q$ *σημεία της μορφής* (x, y), *όπου* $x, y \in \mathbb{F}_q$, *τα οποία ικανοποιούν την εξίσωση της καμπύλης.*

Μπορεί κανείς να καταγράψει τα σημεία της $\mathbb{E}$, *θέτοντας* $x = 0, 1, \cdots, q-1$ *στην εξίσωσή της και βλέποντας αν το* $x^3 + ax + b$ *είναι τετράγωνο* mod$q$. *Γνωρίζουμε ότι μόνο τα μισά από τα στοιχεία του* $\mathbb{F}_q^*$ *έχουν τετραγωνικές ρίζες, οπότε αν τα* $x^3 + ax + b$ *ήταν τυχαία στοιχεία από του σώματος. Επομένως, θα περίμενε κανείς ότι μόνο τα μισά από τα* $2q$ *το πλήθος σημεία του* $\mathbb{F}_q \times \mathbb{F}_q$, *θα ήταν λύσεις της εξίσωσης της* $\mathbb{E}$. *Οπότε αναμένει κανείς να βρει το πολύ* $q+1$ *σημεία.*

Έστω χ *ο τετραγωνικός χαρακτήρας του* $\mathbb{F}_q$. [1] *Πρόκειται για μία απεικόνιση που αντιστοιχίζει το* $x \in \mathbb{F}_q^*$ *στο* ± 1, *ανάλογα με το αν το* x *είναι τετράγωνο στο* $\mathbb{F}_q$ *(και ορίζουμε* $\chi(0) = 0$). *Για παράδειγμα, αν* $q = p$ *είναι πρώτος, τότε το* $\chi(x) = \left(\frac{x}{p}\right)$ *είναι το σύμβολο του Legendre.*

Θέτουμε $u = x^3 + ax + b$. *Επομένως ο στόχος μας είναι να απαριθμήσουμε τα* $y \in \mathbb{F}_q$ *για τα οποία* $y^2 = u$. *Για κάθε τέτοιο* y *το πλήθος των λύσεων για την* $y^2 = u$ *(άρα και για την εξίσωση της* $\mathbb{E}$*) είναι* $\chi(u) + 1$. *Συνολικά δηλαδή, προσμετρώντας και το σημείο στο άπειρο, οι λύσεις είναι*

$$1 + \sum_{x \in \mathbb{F}_q} \big(1 + \chi(x^3 + ax + b)\big) \;=\; q + 1 + \sum_{x \in \mathbb{F}_q} \chi(x^3 + ax + b)\,.$$

Θα περίμενε κανείς ότι το $\chi(x^3 + ax + b)$ *θα ήταν εξίσου πιθανό να είναι* $+1$ *όσο και* -1. *Παίρνοντας το άθροισμα πάνω από όλα τα* $x \in \mathbb{F}_q$ *είναι σα να πραγματοποιούμε το εξής πείραμα: ρίχνουμε ένα νόμισμα, προχωρούμε ένα βήμα μπροστά αν φέρουμε «κεφαλή», ένα βήμα πίσω αν φέρουμε «γράμματα». Στη θεωρία πιθανοτήτων υπολογίζεται ότι η απόσταση (μετρημένη σε βήματα) που διασχίζεται κατόπιν* q *ρίψεων, είναι της τάξης του* $\sqrt{q}$. *Το άθροισμα* $\sum_{x \in \mathbb{F}_q} \chi(x^3 + ax + b)$ *συμπεριφέρεται παρόμοια με το παραπάνω πείραμα. Ειδικότερα, βρίσκει κανείς ότι το άθροισμα αυτό φράσσεται από το* $2\sqrt{q}$. *Αυτό το αποτελεσμα συνιστά το θεώρημα που διατύπωσε ο Helmut Hasse στα 1930.*

[1] Περισσότερα για χαρακτήρες μπορεί κανείς να βρει στον σύνδεσμο http://web-server.math.uoc.gr:1080/proptyxiakes/ptyxiakes/Kouta_PE.pdf.

Θεώρημα 2.4.1 *(**Θεώρημα Hasse**) Έστω ότι η ελλειπτική καμπύλη* $\mathbb{E} \pmod q$ *έχει* N *το πλήθος σημεία. Τότε*

$$|N - (q+1)| \ \leq \ 2\sqrt{q}\,.$$

2.5 Παραδείγματα σε Maple

```
> restart;
```

We define a procedure L in order to verify whether that the points we are about to use, are points on a given elliptic curve E, or not. If this statement is true, we expect to find L=0.

```
> L:= proc(G,P)
>    local  ECeval;
>    ECeval := (P[2])^2 - ((P[1])^3 + G[1]* P[1] + G[2]) mod (G[3]);
>    return ECeval;
> end:
```

The following procedure computes the coefficients of the point C = A + B, over an elliptic curve E

```
> gop := proc(G,A,B)
>   local   a,b,p,xA,yA,xB,yB,xC,yC,C,slope;
>
>   a := G[1];
>   b := G[2];
>   p := G[3];
>
>
> #   The case where  at least one of A and B is the point at infinity
>
> if A = [infinity,infinity] then
>   if L(G,B)<>0 then
>       print(B, `is not a point of the elliptic curve.`);
>       return `computation infeasible`;
>   else
>       return B;
>   fi;
> fi;
>
```

```
>
>  if B = [infinity,infinity] then
>    if L(G,A)<>0 then
>        print(A, `is not a point of the elliptic curve.`);
>        return `computation infeasible`;
>    else
>        return A;
>    fi;
>  fi;
>
>
>
>  xA := A[1];
>  yA := A[2];
>  xB := B[1];
>  yB := B[2];
>
>
>  if L(G,A)<>0 then
>    print(A, `is not a point of the elliptic curve.`);
>    return `computation infeasible`;
>  fi;
>
>
>  if L(G,B)<>0 then
>    print(B, `is not a point of the elliptic curve.`);
>    return `computation infeasible`;
>  fi;
>
>
>
>
>  # The case  where A = B and  yA is not equal to 0
>
>
>  if (xA - xB) modp = 0 and (yA - yB) modp =0 and yA modp <> 0
>  then
>    if gcd(p,2*yA) = 1 then
>        slope := (3*xA^2 + a) / (2*yA) mod p;
>        xC := slope^2 - xA - xB mod p;
>        yC := -yA + slope*(xA - xC) mod p;
```

```
>         C := [xC,yC];
>         return C;
>     else
>         print(‘The modular inverse of the slope’s denominator‘,
>                2*yA, ‘does not exist‘);
>         print(‘Thus, the sum of points ‘, A, B, ‘is not computable‘);
>         return ‘computation infeasible‘;
>     fi;
>
>
>   # The case where either
>   #    xA = xB and yA is not equal to yB or
>   #      A = B and yA is not equal to 0 or
>
>   elif ((xA - xB) mod p = 0 and (yA - yB) mod p <>0) or
>   ((xA - xB) mod p = 0 and yA mod p =0) then
>         C:= [infinity, infinity];
>   else
>
>
>   # The case where xA is not equal to xB
>
>     if gcd(p,xA-xB) = 1 then
>         slope := (yA - yB) / (xA - xB) mod p;
>         xC := slope^2 - xA - xB mod p;
>         yC := -yA + slope*(xA - xC) mod p;
>         C := [xC,yC];
>         return C;
>     else
>         print(‘The modular inverse of the slope’s denominator‘,
>               xA-xB, ‘does not exist‘);
>         print(‘Thus, the sum of points ‘, A, B, ‘is not computable‘);
>         return ‘computation infeasible‘;
>     fi;
>   fi;
>   end:
>
>
```

Vector G provides the feedback for the elliptic curve E we will work with, i.e.

E : $y\hat{}2 = (x\hat{}3 + G[1]^* x + G[2]\ ))\ mod\ (G[3])$

> ***Example 1***
> ***Here we use the elliptic curve E: y^2 = (x^3 + 5x -5) mod(455839)***
>
```
> G := [5,-5,455839]:
> A :=[1, 1]:
> B := [1,213842]:
> C := [1,0]:
> F := [471,20856]:
> Infty := [infinity, infinity]:
>
> AplusA := gop(G,A,A);
```

$$AplusA := [14, 455786]$$

```
>
>
> AplusInfty := gop(G,A,Infty);
> InftyplusA := gop(G, Infty, A);
```

$$AplusInfty := [1, 1]$$

$$InftyplusA := [1, 1]$$

```
>
>
> AplusB := gop(G, A, B);
```

$$AplusB := [\infty, \infty]$$

```
>
>
> AplusC := gop (G, A, C);
```

$$[1, 0], \text{`is not a point of the elliptic curve.`}$$

$$AplusC := \text{`computation infeasible`}$$

```
>
>
> AplusF := gop (G, A, F)·
```

$$AplusC := [390322, 172152]$$

Example 2
Here we use the elliptic curveE: y^2 = (x^3 + 4x +4) mod(2773)

```
> G2 := [4,4,2773]:
> A2 := [1853,0]:
> B2 := [642, 669]:
> C2 := [1468,983]:
> F2 := [591, 1236]:
> F3 := [592, 1236]:
>
> gop(G2, A2, A2);
```

$$[\infty, \infty]$$

```
> gop(G2, F2, C2);
```

$$[1468, 1790]$$

```
>
>
> gop(G2, B2, C2);
```

`The modular inverse of the slope's denominator `, -826, ‘does not exist‘
‘Thus, the sum of points `, $[642, 669]$, $[1468, 983]$, `is not computable‘
`computation infeasible‘

```
> gop(G2, B2, F3);
```

$[592, 1236]$, `is not a point of the elliptic curve.‘
`computation infeasible`

Κεφάλαιο 3

Παραγοντοποίηση Ακέραιου με Χρήση Ελλειπτικών Καμπύλων

Στα μέσα της δεκαετίας του 1980, ο Hendrik Lenstra έδωσε νέα ώθηση στην μελέτη των ελλειπτικών καμπύλων, αναπτύσσοντας έναν αποδοτικό αλγόριθμο παραγοντοποίησης θετικών ακεραίων, ο οποίος χρησιμοποιούσε ελλειπτικές καμπύλες. Ο αλγόριθμος αυτός φάνηκε στην πορεία ότι είναι ιδιαίτερα αποτελεσματικός για την παραγοντοποίηση αριθμών με περί τα 60 ψηφία, ενώ, για μεγαλύτερους αριθμούς, χρησιμεύει στην εύρεση πρώτων παραγόντων, οι οποίοι έχουν από 20 εώς και 30 ψηφία.

3.1 Ο $p-1$ Αλγόριθμος Παραγοντοποίησης του Pollard

Ορισμός 3.1.1 *Ένας ακέραιος m είναι ομαλός ως προς το B, ή απλούστερα B - ομαλός, εάν και μόνο εάν όλοι οι πρώτοι παράγοντες του m είναι μικρότεροι ή ίσοι προς το B. Σε αυτή την περίπτωση, το άνω φράγμα B λέγεται φράγμα ομαλότητας.*

3.1.1 Περιγραφή της $p-1$ Μεθόδου του Pollard

Έστω ότι επιθυμούμε να παραγοντοποιήσουμε τον σύνθετο $n \in \mathbb{N}$, και ότι έχει επιλεχθεί ένα φράγμα ομαλότητας B.

Ας υποθέσουμε ότι ο p είναι κάποιος (άγνωστος μέχρι στιγμής) πρώτος παράγοντας του n. Αν ο p έχει την ιδιότητα ότι ο $p-1$ δεν έχει κανέναν μεγάλο πρώτο παράγοντα, τότε η ακόλουθη μέθοδος είναι ουσιαστικά βέβαιο ότι θα οδηγήσει σε εύρεση του p.

1. *Επιλέγουμε μία βάση* $a \in \mathbb{N}$ *όπου* $2 \leq a < n$. *Για παράδειγμα, το* a *θα μπορούσε να είναι ίσο με 2, ή 3, ή με οποιονδήποτε τυχαία επιλεγμένο θετικό ακέραιο μικρότερο του* n. *Υπολογίζουμε το* $g = \mu\kappa\delta(a, n)$. *Αν* $g > 1$, *τότε έχουμε βρει έναν παράγοντα του* n. *Αλλιώς, συνεχίζουμε στο βήμα 2.*

2. *Επιλέγουμε έναν ακέραιο* k *τέτοιο ώστε να είναι πολλαπλάσιο όλων ή των περισσότερων από τους ακέραιους που είναι μικρότεροι από το* B. *Για παράδειγμα, το* k *μπορεί να είναι το* $B!$, *ή μπορεί να είναι το ελάχιστο κοινό πολλαπλάσιο όλων των ακέραιων* $\leq B$.

3. *Υπολογίζουμε*

$$a_1 \equiv a^k \pmod{n}, \text{ και } \mu\kappa\delta(a_1 - 1, n).$$

Σε αυτό το σημείο θα πρέπει να σημειωθεί ότι, για την καλύτερη διαχείριση των διαθέσιμων πόρων (μνήμη υπολογιστή), δεν υπολογίζουμε το a^k *και μετά να το ανάγουμε* mod$n$. *Αντ' αυτού, αν, για παράδειγμα, χρησιμοποιήσουμε* $k = B!$, *υπολογίζουμε το* $a^{B!} \bmod n$ *αναδρομικά, μέσω της* $a^{b!} \equiv \left(a^{(b-1)!}\right)^b \pmod{n}$, *για* $b = 2, 3, \cdots, B$. *Ανάλογα, για κάθε άλλη επιλογή του* k, *αναζητούμε κάποιον υπολογιστικά οικονομικότερο τρόπο για την εύρεση του το* $a^k \pmod{n}$.
Η μέθοδος που θα προτιμήσουμε, είναι να γράψουμε το $B!$ *στη δυαδική του αναπαράσταση και να υπολογίσουμε το modular εκθετικό με τη μέθοδο του επαναλαμβανόμενου τετραγωνισμού (βλ. Λήμμα 3.1.2).*

4. *Υπολογίζουμε το* $d = \mu\kappa\delta(a^k - 1, n)$, *χρησιμοποιώντας τον αλγόριθμο του Ευκλείδη και το ισοϋπόλοιπο του* $a^k \pmod{n}$ *από το προηγούμενο βήμα.*

5. *Αν* $d > 1$, *τότε έχουμε έναν παράγοντα του* n. *Αλλιώς, αν* d *δεν είναι μη τετριμμένος παράγοντας του* n, *ο αλγόριθμος αποτυγχάνει. Οπότε, επαναλαμβάνουμε τη μέθοδο κάνοντας μία νέα επιλογή για το* a *ή /και μία νέα επιλογή για το* k.

Λήμμα 3.1.2 **Μέθοδος Επαναλαμβανόμενου Τετραγωνισμού (Repeated Squaring Method).** *Δεδομένων* $b, r, n \in N$, *ας υποθέσουμε ότι επιθυμούμε να υπολογίσουμε την* r *δύναμη του* b mod$n$ *με τη μέθοδο του επαναλαμβανόμενου τετραγωνισμού. Το εγχείρημα αυτό αποτελεί μία υπολογιστική πρόκληση, ειδικά όταν και οι δύο εκ των* r *και* n *είναι πολύ μεγάλοι. Η μέθοδος αυτή είναι αρκετά εξυπνότερη από ότι ο επαναλαμβανόμενος πολλαπλασιασμός του* b *με τον εαυτό του.*

Στα ακόλουθα θεωρούμε ότι $b < n$,[1] *και ότι όποτε κάνουμε έναν πολλαπλασιασμό, παίρνουμε το ελάχιστο θετικό ισοϋπόλοιπο του γινομένου* mod$n$. *Κατ' αυτόν*

[1] Αν $b \geq n$, τότε εργαζόμαστε με χρήση του b', όπου $b' \equiv b \pmod{n}$.

τον τρόπο δεν ερχόμαστε ποτέ αντιμέτωποι με ακέραιο μεγαλύτερο του n^2. Ακολουθεί η περιγραφή του αλγορίθμου.

Χρησιμοποιούμε το a για να συμβολίσουμε το μερικό γινόμενο. Όταν τελειώσουμε, το a θα είναι ίσο με το ελάχιστο θετικό ισοϋπόλοιπο του $b^r \pmod n$. Ξεκινάμε θέτοντας $a = 1$. Έστω ότι τα $r_0, r_1, \cdots, r_{k-1}$ συμβολίζουν τα ψηφία του r στην δυαδική του αναπαράσταση, δηλαδή

$$r \;=\; r_0 + 2r_1 + 4r_2 + \cdots + 2^{k-1}r_{k-1}\ .$$

Καθένα εκ των r_j είναι ίσο είτε με 0 είτε με 1. Αν $r_0 = 0$, τότε παραμένει $a = 1$. Αλλιώς, αν $r_0 = 1$ τότε $a = b$. Υπολογίζουμε το b^2 και θέτουμε $b_1 \equiv b^2 \pmod n$. Αν $r_1 = 0$, το a μένει αμετάβλητο. Αν $r_1 = 1$, πολλαπλασιάζουμε το a επί b_1 και η νέα τιμή του a είναι το ελάχιστο θετικό ισοϋπόλοιπο του $ab_1 \pmod n$. Σε επόμενο βήμα, τετραγωνίζουμε το b_1 και θέτουμε b_2 να είναι το ελάχιστο θετικό ισοϋπόλοιπο του $b_1^2 \pmod n$. Εάν $r_2 = 1$, πολλαπλασιάζουμε το a επί b_2, αλλιώς το a παραμένει ως έχει. Συνεχίζουμε κατ' αυτόν τον τρόπο. Στο $j-$ οστό βήμα ($j = 0, 1, \cdots, k-1$), έχουμε υπολογίσει το $b_j \equiv b^{2^j} \pmod n$. Αν $r_j = 1$, δηλαδή αν το 2^j εμφανίζεται στη διαδική αναπαράσταση του r, τότε συμπεριλαμβάνουμε το b_j στο γινόμενο του a (αν το 2^j απουσιάζει από το r, τότε όχι). Είναι τώρα απλό να δει κανείς ότι μετά το $(k-1)-$ οστό βήμα έχουμε το επιθυμητό $a \equiv b^r \pmod n$.

Ακολούθως δίνουμε ένα παράδειγμα εφαρμογής του αλγόριθμου επαναλαμβανόμενου τετραγωνισμού:

Calculation of the least non negative residue of 3^(61) modulo 101 by use of the successive squaring method

```
> restart:
> with(Bits);
```

[*And*, *FirstNonzeroBit*, *GetBits*, *Iff*, *Implies*, *Join*, *Nand*, *Nor*, *Not*, *Or*, *Settings*, *Split*, *String*, *Xor*]

```
> n := 101;
> b:=3:
> b:= b mod(n);
> r:= 61;
```

$$n := 101$$
$$b := 3$$
$$r := 61$$

we now produce the binary representation of the exponentp.

R denotes the vector [r_1, r_1,...,r_k], where r=Sum_(j=1)^(k) r_j 2^j

```
> convert(r,binary);
> R:= Split(r);
> k:= nops(R);
```

$$R := \begin{matrix} 111101 \\ [1,0,1,1,1,1] \end{matrix}$$
$$k := 6$$

we symbolize by *part_prod* ***the partial product that the method generates at each of the k steps.***

```
> part_prod:=1;
> B:=b;
> if R[1]=1 then
> part_prod:=part_prod*B;
> fi;
```

$$part\_prod := 1$$
$$B := 3$$
$$part\_prod := 3$$

```
> for i from 2 to k do
> B:= B^2 mod(n);
> if R[i]=1 then
> part_prod:=part_prod * B mod(n);
> fi;
> od;
> part_prod;
```

$$B := 9$$
$$B := 81$$
$$B := 97$$
$$B := 16$$
$$B := 54$$
$$7$$

```
> b^r mod(n);
```

$$7$$

3.1.2 Ανάλυση της $p-1$ Μεθόδου του Pollard

Ο αλγόριθμος του Pollard βασίζεται στον ακόλουθο συλλογισμό. Ας υποθέσουμε ότι το k διαιρείται από όλους τους θετικούς ακέραιους $\leq B$, και επιπλέον ας υποθέσουμε ότι ο p είναι ένας πρώτος διαιρέτης του n, τέτοιος ώστε ο $p-1$ να είναι γινόμενο μικρών πρώτων δυνάμεων, οι οποίες είναι όλες μικρότερες από B. Έπεται ότι το k είναι πολλαπλάσιο του $p-1$ (αφού είναι πολλαπλάσιο όλων των δυνάμεων πρώτων που εμφανίζονται στην παραγοντοποίηση του $p-1$). Συνεπώς, από το Μικρό Θεώρημα του Fermat, έχουμε $a^k \equiv 1 \pmod{p}$. Τότε, $p \mid \mu\kappa\delta(a^k-1, n)$. Έτσι, η

μοναδική περίπτωση κατά την οποία μπορεί στο βήμα 4 να αποτύχουμε να βρούμε μη τετριμμένο παράγοντα του n, είναι η πείπτωση κατά την οποία $a^k \equiv 1 \pmod{n}$.

Παράδειγμα 3.1.3 *Χρησιμοποιούμε την μέθοδο του Pollard για να παραγοντοποιήσουμε το $n = 540143$. Επιλέγουμε $a = 2$ και $B = 8$ (άρα $k = \epsilon\kappa\pi(1, 2, \cdots, 8) = 840$). Με τη μέθοδο του επαναλαμβανόμενου modular τετραγωνισμού, βρίσκουμε ότι $2^{840} \equiv 53047 \pmod{n}$. Επιπλέον, $\mu\kappa\delta(53046, n) = 421$. Αυτό οδηγεί στην παραγοντοποίηση $540143 = 421 \cdot 1283$.*

Η βασική αδυναμία της μεθόδου του Pollard είναι εμφανής εάν επιχειρήσουμε να την χρησιμοποιήσουμε σε περίπτωση όπου για κάθε πρώτο διαιρέτη p του n, ο $p - 1$ διαιρείται από έναν σχετικά μεγάλο πρώτο (ή δύναμη πρώτου).

Παράδειγμα 3.1.4 *Έστω $n = 491389$. Σε αυτή την περίπτωση, δε θα βρίσκαμε μη τετριμένο παράγοντα, ώσπου να επιλέξουμε $B \geq 191$, αφού ο n παραγοντοποιείται ως $n = 383 \cdot 1283$. Έχουμε $383 - 1 = 2 \cdot 191$ και $1283 - 1 = 2 \cdot 641$, όπου οι 191 και 641 είναι και οι δύο πρώτοι. Επιπλέον, εκτός από $a \equiv 0, \pm 1 \pmod{383}$ όλα τα υπόλοιπα a έχουν τάξη modulo 383 είτε 191 είτε 382. Όπως και εκτός από $a \equiv 0, \pm 1 \pmod{1283}$ όλα τα υπόλοιπα a έχουν τάξη modulo 1283 είτε 641 είτε 1282. Επομένως, εκτός κι αν το k που επιλέγουμε διαιρείται από το 191 (ή το 641), στο βήμα 4 θα παίρνουμε συνεχώς $\mu\kappa\delta(a^k - 1, n) = 1$.*

Το βασικό μειονέκτημα της μεθόδου $p - 1$ του Pollard είναι ότι οι ελπίδες μας εξαρτώνται από την ομάδα $(\mathbb{Z}/p\mathbb{Z})^*$. Πιο συγκεκριμένα, από τις ομάδες αυτές καθώς ο p διατρέχει το σύνολο των πρώτων παραγόντων του n. Για συγκεκριμένο n, οι ομάδες αυτές είναι συγκεκριμένες. Αν τυχαίνει όλες να έχουν τάξη διαιρετή από έναν μεγάλο πρώτο, η μέθοδος δεν αποδίδει.

Η ουσιαστική διαφορά με τον αλγόριθμο του Lenstra που θα μελετήσουμε έγκειται στο γεγονός ότι δουλεύοντας πάνω από σώματα $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, έχουμε δυνατότητα να χρησιμοποιήσουμε πληθώρα ομάδων, και μπορούμε ρεαλιστικά να ελπίζουμε ότι θα βρούμε μία ομάδα η τάξη της οποίας δεν διαιρείται από κάποιον μεγάλο πρώτο ή από δύναμη πρώτου.

3.1.3 Εφαρμογή της Μεθόδου του Pollard σε Maple

Για το ακόλουθο κάνουμε χρήση της μεθόδου του επαναλαμβανόμενου τετραγωνισμού, την οποία παρουσιάσαμε στο Λήμμα 3.1.2.

Factoring $n = 26869$ by use of Pollard's p - 1 Algorithm. To begin with, we select a base a and a smoothness bound B .

```
> restart:
> with(Bits);
```

$$[And, FirstNonzeroBit, GetBits, Iff, Implies, Join, Nand, Nor, Not, Or, Settings, Split, String, Xor]$$

```
> n := 26869;
> a := 7;
> B := 13;
> g := gcd(a, n);
```

$$n := 26869$$
$$a := 7$$
$$B := 13$$
$$g := 1$$

we now select an integer k ***and perform the 3rd srep of the algorithm, using the successive squaring method in order to compute*** $a\hat{}k \pmod{n}$.

```
> k := B!;
```

$$k := 6227020800$$

```
> convert(k,binary):
> R := Split(k);
> m := nops(R);
> part_prod := 1:
> A := a:
> if R[1] = 1 then
> part_prod := part_prod * A:
> fi;
> for i from 2 to m do
> A := A^2 mod(n):
> if R[i] = 1 then
> part_prod := part_prod * A mod(n):
> fi:
> od:
> part_prod;
```

$$R := [0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,1,0,1,0,0,1,1,0,0,1,1,1,0,1]$$
$$m := 33$$
$$24445$$

Verifying the outcome of the algorithm:

```
> a_1 := gcd(part_prod -1, n);
> a_2 := n/a_1;
> a_1 * a_2;
```

$$a\_1 := 97$$

$a\_2 := 277$
26869

3.2 Παραγοντοποίηση ακεραίων με χρήση ελλειπτικών καμπύλων

Εάν επιθυμούμε να παραγοντοποιήσουμε τον θετικό ακέραιο n, για τον οποίο γνωρίζουμε ότι είναι γινόμενο δύο πρώτων αριθμών. Επιλέγουμε τυχαία ελλειπτική καμπύλη $\mathbb{E} \bmod n$ και σημείο P πάνω στην $\mathbb{E}$. Πρακτικά, επιλέγει κανείς πολλαπλές (περίπου 14 για αριθμούς περί τα 50 ψηφία, περισσότερες για μεγαλύτερους ακέραιους) καμπύλες και αντίστοιχα σημεία επ' αυτών και τρέχουμε τον αλγόριθμο παράλληλα.

3.2.1 Ο Αλγόριθμος

Έστω λοιπόν n ένας θετικός περιττός ακέραιος. Περιγράφουμε την πιθανοτική μέθοδο του Lenstra για την παραγοντοποίηση του n.

Τίθεται το πρακτικό ερώτημα, πώς επιλέγουμε την ελλειπτική καμπύλη $\mathbb{E}$ που θα χρησιμοποιήσουμε; Πρώτα επιλέγουμε σημείο $P(x_0, y_0)$ και μία σταθερά a. Κατόπιν, υπολογίζουμε τη σταθερά b, έτσι ώστε οι συντεταγμένες του P να ικανοποιούν την εξίσωση Weierstrass (2.1). Αυτή η μέθοδος είναι αποδοτικότερη από το να κανείς τα a και b και να επιδιώξει να βρει σημείο P που να ανήκει στην ελλειπτική καμπύλη με εξίσωση $y^2 = x^3 + ax + b$.

Υποθέτουμε τώρα ότι έχουμε παράγει ένα ζεύγος $(\mathbb{E}, P)$ που αποτελείται από μία ελλειπτική καμπύλη με εξίσωση της μορφής $y^2 = x^3 + ax + b$ με $a, b \in \mathbb{Z}$ και ένα σημείο $P = (x, y) \in \mathbb{E}$ και συνεχίζουμε με τη μέθοδο που θα περιγράψουμε στη συνέχεια. Αν η μέθοδος αποτύχει να δώσει έναν μη τετριμμένο παράγοντα του n, τότε παράγουμε ένα νέο ζεύγος $(\mathbb{E}, P)$ και επαναλαμβάνουμε τη διαδικασία.

Πριν να ξεκινήσουμε να δουλεύουμε με την $\mathbb{E} \bmod n$ που έχουμε επιλέξει, πρέπει να επιβεβαιώσουμε ότι είναι όντως ελλειπτική καμπύλη $\bmod p$, για κάθε πρώτο διαιρέτη p του n, δηλαδή ότι το κυβικό πολυώνυμο στο δεξί μέλος της εξίσωσης της καμπύλης έχει διακεκριμένες ρίζες $\bmod p$. Αυτό ισχύει εάν και μόνο εάν η διακρίνουσα του κυβικού πολυωνύμου είναι μη μηδενική $\bmod p$ για κάθε $p \mid n$. Άρα

πρέπει και αρκεί η διακρίνουσα $4a^3+27b^2$ να είναι πρώτη προς το n. Οπότε ελέγχουμε εάν $\mu\kappa\delta(4a^3+27b^2, n) = 1$ και αν ναι, μπορούμε να συνεχίσουμε. Φυσικά, αν αυτή η διακρίνουσα είναι γνήσια μεταξύ των 1 και n, έχουμε έναν μη τετριμμένο διαιρέτη του n οπότε έχουμε τελειώσει. Αλλιώς, αν αυτός ο μέγιστος κοινός διαιρέτης είναι ίσος με n, θα πρέπει να επιλέξουμε άλλη ελλειπτική καμπύλη για να συνεχίσουμε.

Σε δεύτερο βήμα, επιλέγουμε δύο ακέραια φράγματα ομαλότητας, δηλαδή δύο ακέραιους B (ίσως κοντά στο 10^8) και C. Εδώ το B είναι φράγμα για τους πρώτους διαιρέτες του αριθμού k επί τον οποίο πολλαπλασιάζουμε το σημείο P. Αν το B είναι μεγάλο, τότε είναι μεγάλη και η πιθανότητα το ζεύγος μας $(\mathbb{E}, P)$ να έχει την ιδιότητα $kP \equiv \infty \bmod p$ για κάποιο p με $p \mid n$. Από την άλλη πλευρά, όσο μεγαλύτερο είναι το B τόσο περισσότερος χρόνος θα χρειαστεί για να υπολογιστεί το $kP \bmod p$. Επομένως, η επιλογή του B θα πρέπει να γίνει με τέτοιο τρόπο ώστε να αναμένεται να ελαχιστοποιηθεί ο χρόνος εκτέλεσης του υπολογισμού. Όσο αφορά στο C, αυτό είναι ένα φράγμα για τους πρώτους διαιρέτες p του n για τους οποίους αναμένουμε να πάρουμε μία σχέση $kP \bmod p \equiv \infty \bmod p$. Τότε, επιλέγουμε k ώστε να δίνεται από τη σχέση

$$k \;=\; \prod_{\ell \le B} \ell^{\alpha_\ell},$$

όπου, για κάθε πρώτο ℓ με $\ell \le B$, ο $\alpha_\ell = [\log C / \log \ell]$ είναι ο μέγιστος εκθέτης ώστε $\ell^{\alpha_\ell} \le C$. Αυτό σημαίνει ότι το k είναι το γινόμενο όλων των πρώτων δυνάμεων $\le C$ που είναι δυνάμεις πρώτων $\le B$.

Τότε, το θεώρημα 2.4.1 μας λέει ότι, αν ο p είναι τέτοιος ώστε $p+1+2\sqrt{p} < C$ και η τάξη της ομάδας των σημείων της $\mathbb{E} \bmod p$ δεν είναι διαιρετή από κάποιον πρώτο $> B$, τότε το k είναι πολλαπλάσιο αυτής της τάξης, οπότε $kP \equiv \infty \bmod p$.

Παράδειγμα 3.2.1 Ας υποθέσουμε ότι έχουμε επιλέξει $B = 20$, και επιθυμούμε να παραγοντοποιήσουμε έναν δεκαψήφιο ακέραιο n, ο οποίος μάλλον είναι γινόμενο δύο πενταψήφιων πρώτων παραγόντων (με αυτό εννοούμε ότι οι πρώτοι παράγοντες του n είναι περίπου της ίδιας τάξης μεγέθους). Τότε, επιλέγουμε $C = 100700$ και $k = 2^{16} \cdot 3^{10} \cdot 5^7 \cdot 7^5 \cdot 11^4 \cdot 13^4 \cdot 17^4 \cdot 19^3$. □

Επιστρέφουμε τώρα στην περιγραφή του αλγόριθμου. Εργαζόμενοι $\bmod n$ επιχειρούμε να υπολογίσουμε το kP ως εξής: $2P, 2(2P), 2(4P), \cdots, 2^{\alpha_2}P$, μετά $3(2^{\alpha_2}P), 3(3\,2^{\alpha_2}P)$, $\cdots$, $3^{\alpha_3}2^{\alpha_2}P$ κόκ, ώσπου τελικά να υπολογιστεί το $\prod_{\ell \le B} \ell^{\alpha_\ell} P$. Αυτό υλοποιείται με τη μέθοδο επαναλαμβανόμενου διπλασιασμού, η οποία περιγράφεται παρακάτω, στο Λήμμα 3.2.3. Σε αυτούς τους υπολογισμούς, όποτε χρειάζεται να γίνει διαίρεση modulo n, χρησιμοποιούμε τον Ευκλείδειο Αλγόριθμο για να βρούμε το πολλαπλασιαστικό αντίστροφο $\bmod n$. Αν σε οποιοδήποτε σημείο ο αλγόριθμος αποτύχει μα

δώσει το αντίστροφο, τότε είτε βρίσκουμε έναν μη τετριμμένο διαιρέτη του n ή τον ίδιο τον n, μέσω του μέγιστου κοινού διαιρέτη του n και του παρονομαστή. Στην πρώτη περίπτωση, ο στόχος μας επιτυγχάνεται, και ο αλγόριθμος ολοκληρώνεται επιτυχώς. Στη δεύτερη περίπτωση, ο αλγόριθμός μας αποτυγχάνει και θα πρέπει να επιλέξουμε άλλο ζεύγος $(\mathbb{E}, P)$. Αν ο αλγόριθμος του Ευκλείδη διαρκώς δίνει έναν αντίστροφο, οπότε το $kP \bmod n$ πράγματι υπολογίζεται, τότε θα πρέπει, και σε αυτή την περίπτωση να επιλέξουμε άλλο ζεύγος $(\mathbb{E}, P)$.

Αυτό ολοκληρώνει την περιγραφή του αλγόριθμου παραγοντοποίησης ακεραίων με χρήση ελλειπτικών καμπύλων.

Παράδειγμα 3.2.2 Έστω $n = 2773$. Θεωρούμε το $P = (1, 3)$ και επιλέγουμε $a = 4$. Επειδή θέλουμε να ισχύει $3^2 \equiv 1^3 + 4 \cdot 1 + b \bmod 2773$, παίρνουμε $b = 4$. Επομένως, η καμπύλη $\mathbb{E}$ με την οποία θα δουλέψουμε είναι η εξής.

$$\mathbb{E}: \quad y^2 \equiv x^3 + 4x + 4 \pmod{2773}.$$

Υπολογίζουμε το $2P = (1771, 705)$, με χρήση των τύπων που αποδείξαμε στο Κεφάλαιο 2, όπου για την εύρεση του m χρειάζεται να υπολογίσουμε το $(2 \cdot y_0)^{-1} \bmod 2773$. Προϋπόθεση για την αντιστροφή του $6 \pmod{2773}$ είναι ο $\mu\kappa\delta(6, 2773)$ να είναι ίσος με 1, που ισχύει. Οπότε χρησιμοποιούμε τον ευκλείδειο αλγόριθμο για τον υπολογισμό.

Ας υπολογίσουμε τώρα το σημείο $3P = 2P + P$. Η ευθεία που διέρχεται από τα σημεία $2P = (1771, 705)$ και $P = (1, 3)$ έχει συντελεστή διεύθυνσης $702/1770$. Στην προσπάθεια να αντιστρέψουμε το $1770 \bmod 2773$, βρίσκουμε

$$\begin{aligned} 2773 &= 1 \cdot 1770 + 1003 \\ 1770 &= 1 \cdot 1003 + 767 \\ 1003 &= 1 \cdot 767 + 236 \\ 767 &= 2 \cdot 236 + 59 \\ 236 &= 4 \cdot 59 + 0, \end{aligned}$$

απ' όπου συμπεραίνουμε ότι $\mu\kappa\delta(1770, 2773) = 59$. Η διαπίστωση ότι $\mu\kappa\delta(1770, 2773) \neq 1$, σημαίνει ότι το 1770 δεν είναι αντιστρέψιμο $\bmod 2773$. Οπότε τι κάνουμε τώρα; Ο αρχικός μας στόχος ήταν να παραγοντοποιήσουμε το 2773, άρα δεν χρειάζεται να κάνουμε τίποτε παραπάνω. Βρήκαμε τον ένα παράγοντά του, το 59, κι έτσι παίρνουμε την παραγοντοποίηση $2773 = 59 \cdot 47$. □

Ας δούμε τι συμβαίνει. Με βάση το Κινέζικο Θεώρημα Υπολοίπων, μπορούμε να δούμε την $\mathbb{E}$ ως ζεύγος ελλειπτικών καμπύλων, την $\mathbb{E} \bmod 59$ και την $\mathbb{E} \bmod 47$ και διαπιστώνουμε ότι $3P = \infty \bmod 59$, ενώ είναι $3P \neq \infty \bmod 47$. Επομένως,

όταν επιχειρήσαμε να υπολογίσουμε τις συντεταγμένες του σημείου $3P$*, συναντήσαμε έναν συντελεστή διεύθυνσης που ήταν άπειρος* $\bmod 59$ *αλλά πεπερασμένος* $\bmod 47$*. Με άλλα λόγια, είχαμε έναν μηδενικό παρονομαστή* $\bmod 59$*, αλλά μη μηδενικό* $\bmod 47$*, γεγονός που μας επέτρεψε, υπολογίζοντας τον μέγιστο κοινό διαιρέτη, να απομονώσουμε τον παράγοντα* 59*.*

Αυτού του τύπου η ιδέα συνιστά τη βάση για πολλούς αλγόριθμους παραγοντοποίησης. Αν $n = pq$*, δεν είναι εφικτό να απομονώσουμε τους παράγοντες* p *και* q *όσο εκείνοι συμπεριφέρονται όμοια. Ωστόσο, αν βρεθεί κάτι που τους κάνει να συμπεριφέρονται έστω και ελαφρά διαφορετικά, ο στόχος υλοποιείται. Εν προκειμένω, στο παράδειγμα 3.2.2, τα πολλαπλάσια του* P *έφτασαν στο σημείο* ∞*, ταχύτερα* $\bmod 59$ *από ότι* $\bmod 47$*. Εφόσον εν γένει οι πρώτοι* p *και* q *αναμένεται*[2] *να συμπεριφέρονται αρκετά διαφορετικά, θα περίμενε κανείς ότι για τις περισσότερες ελλειπτικές καμπύλες* $\mathbb{E} \pmod{pq}$ *και τα περισσότερα σημεία* P*, τα πολλαπλάσια του* P *δεν θα φτάνουν στο σημείο* $\infty \bmod p$ *και* $\bmod q$ *ταυτόχρονα. Κατα συνέπεια, ένας υπολογισμός μέγιστου κοινού διαιρέτη επιστρέφει είτε το* p *είτε το* q*.*

Συνήθως χρειάζονται αρκετά περισσότερα από τρία ή τέσσερα βήματα για να πάρουμε το σημείο $\infty \bmod p$ *ή* $\bmod q$*. Στην πράξη, χρειάζεται να πολλαπλασιάσει κανείς το σημείο* P *επί έναν μεγάλο αριθμό* B *με πολλούς μικρούς πρώτους παράγοντες, για παράδειγμα,* $B = 10000!$*. Εδώ θα πρέπει να σημειωθεί ότι δεν συνιστάται να επιχειρίσει κανείς να υπολογίσει το σημείο* $B \cdot P$ *με διαδοχικές προσθέσεις* $P + P$*,* $3P = 2P + P$*,* $4P = 3P + P$ *κ.ο.κ., καθώς αυτό θα ήταν ιδιαίτερα χρονοβόρο. Είναι προτιμότερο ο πολλαπλασιασμός* $B \cdot P$ *να εκτελεστεί με τη μέθοδο του επαναλαμβανόμενου διπλασιασμού (το ανάλογο της μεθόδου του επαναλαμβανόμενου τετραγωνισμού 3.1.2), την οποία θα παρουσιάσουμε και θα αποδείξουμε στη συνέχεια (Λήμμα 3.2.3). Η ελπίδα είναι ότι αυτό το πολλαπλάσιο είναι το* ∞ *είτε* $\bmod p$ *ή* $\bmod q$*.*

Παρατηρούμε ότι αυτό είναι το ανάλογο της $p-1$ *μεθόδου παραγοντοποίησης του Pollard. Ωστόσο, υπενθυμίζεται ότι η τελευταία, δεν δίνει αποτέλεσμα όταν ο* $p - 1$ *έχει τουλάχιστον έναν μεγάλο πρώτο παράγοντα. Ο ίδιος τύπος κωλύματος μπορεί να προκύψει με τη μέθοδο των ελλειπτικών καμπύλων την οποία μόλις παρουσιάσαμε, όταν ο φυσικός αριθμός* k *που είναι τέτοιος ώστε* $kP = \infty$ *έχει τουλάχιστον έναν μεγάλο πρώτο παράγοντα, ο οπότε η μέθοδος αποτυγχάνει να επιστρέψει έναν παράγοντα του* n*. Σε αυτή την περίπτωση, απλώς εκτελούμε τον αλγόριθμο αλλάζοντας την επιλογή ελλειπτικής καμπύλης* $\mathbb{E}$*. Η νέα καμπύλη θα είναι ανεξάρτητη από την προηγούμενη και η τιμή* k' *ώστε* $k'P = \infty$*, για τη νέα επιλογή του σημείου* P*, δε θα πρέπει ουσιαστικά να έχει καμία σχέση με την προηγούμενη τιμή του* k*. Έπειτα από αρκετές δοκιμές (ή έπειτα από αρκετές καμπύλες στην περίπτωση όπου ο αλγόριθμος εκτελείται παράλληλα για ορισμένες καμπύλες* $\mathbb{E}_i \bmod n$ *και αντίστοιχα σημεία τους* P_i*) συχνά βρίσκεται κατάλληλη καμπύλη και ο ακέραιος* $n = pq$ *παραγοντοποιείται. Αντίθετα, αν ο* $p-1$ *αλγόριθμος αποτύχει, τότε δεν υπάρχει κάτι που να μπορεί να αλλαχθεί εκτός από την ίδια τη μέθοδο παραγοντοποίησης.*

[2] χάρη στην τυχαιότητα της επιλογής τους.

Λήμμα 3.2.3 **Μέθοδος επαναλαμβανόμενου διπλασιασμού** *(Successive doubling method). Έστω $x = b_1 b_2 \cdots b_w$ ένας ακέραιος αριθμός, γραμμένος σε διαδική μορφή. Έστω P σημείο πάνω στην ελλειπτική καμπύλη $\mathbb{E}$.*

1. *Αρχίζουμε με $k = 1$ και $S_1 = \infty$.*
2. *Αν $b_k = 1$, τότε $R_k = S_k + P$.*
   Αν $b_k = 0$, τότε $R_k = S_k$.
3. *Έστω $S_{k+1} = 2R_k$.*
4. *Αν $k = w$, σταματάμε. Αν $k < w$, προσθέτουμε 1 στο k και συνεχίζουμε στο βήμα (2).*

Όταν ο αλγόριθμος τερματίσει, επιστρέφει το R_w, όπου $R_w = xP$.

Απόδειξη. *Με τη σύμβαση ότι $0 \cdot P = \infty$, έχω ότι*

$$R_1 \;=\; S_1 + b_1 P \;=\; \infty + b_1 P \;=\; b_1 P\,.$$

και επειδή για κάθε k, με $1 < k \leq w$, ισχύει $R_k = S_k + b_k P$ και $S_k = 2R_{k-1}$, έχουμε

$$R_k \;=\; 2R_{k-1} + b_k P\,.$$

Οπότε είναι

$$\begin{aligned}
R_w &= 2R_{w-1} + b_w P \;=\; 2(2R_{w-2} + b_{w-1}P) + b_w P \;=\; 2^2 R_{w-2} + 2^1 b_{w-1} P + b_w P \\
&= 2^2(2R_{w-3} + b_{w-2}P) + 2^1 b_{w-1} P + b_w P \;=\; 2^3 R_{w-3} + 2^2 b_{w-2} P + 2^1 b_{w-1} P + b_w P \\
&\;\;\vdots \\
&= 2^{w-1} b_1 P + 2^{w-2} b_2 P + \cdots + 2^2 b_{w-2} P + 2^1 b_{w-1} P + 2^0 b_w P \\
&= \left(2^{w-1} b_1 + 2^{w-2} b_2 + \cdots + 2^2 b_{w-2} + 2^1 b_{w-1} + 2^0 b_w\right) P \\
&= x\,P
\end{aligned}$$

□

Παράδειγμα 3.2.4 *Έστω ότι επιθυμούμε να παραγοντοποιήσουμε το $n = 455839$. Επιλέγουμε*

$$\mathbb{E} : y^2 \;\equiv\; x^3 + 5x - 5 \quad \text{και} \quad P = (1, 1)\,.$$

Ας υποθέσουμε ότι επιδιώκουμε να υπολογίσουμε το $10!P$. *Υπάρχουν διάφοροι τρόποι για να υλοποιηθεί αυτό. Ένας είναι ο αναδρομικός υπολογισμός* $2!P$, $3!P = 3(2!P)$, $4!P = 4(3!P)$, $\cdots$. *Δηλαδή, έχοντας χρησιμοποιήσει το Λήμμα 3.2.3 για να υπολογίσουμε το* $2!P$, *οπότε έχουμε πάρει κάποιο σημείο* Q *της* $\mathbb{E}$, *υπολογίζουμε το* $3!P$ *ως* $3(2!P) = 3Q$, *με χρήση και πάλι της μεθόδου επαναλαμβανόμενου διπλασιασμού.*

Αν το κάνουμε αυτό, όλα βαίνουν καλώς μέχρι και το $7!P$. *Για την εύρεση όμως του* $8!P$ *χρειάζεται να αντιστρέψουμε το* $599 \bmod 455839$. *Υπολογίζοντας τον* $\mu\kappa\delta(599, 455839)$ *βρίσκουμε* 599, *οπότε παραγοντοποιούμε το* n *ως* $599 \cdot 761$.

Ας εξετάσουμε το εξαγόμενο περεταίρω. Αν από το 2.4.1, μπούμε στη διαδικασία να υπολογίσουμε το πλήθος των σημείων της $\mathbb{E} \bmod 599$ *θα δούμε ότι είναι* $640 = 2^7 \cdot 5$ *και ότι το πλήθος των σημείων της* $\mathbb{E} \bmod 761$ *είναι* $777 = 3 \cdot 7 \cdot 37$. *Επιπλέον, το 640 είναι ο μικρότερος θετικός ακέραιος* m *τέτοιος ώστε* $mP = \infty \bmod 599$ *στην* $\mathbb{E}$, *και το 777 είναι ο μικρότερος θετικός ακέραιος* m *τέτοιος ώστε* $mP = \infty \bmod 777$ *στην* $\mathbb{E}$. *Παρατηρούμε τώρα ότι το* $8!$ *είναι πολλαπλάσιο του 640. Άρα είναι απλό να δει κανείς ότι* $8!P = \infty$ *στην* $\mathbb{E} \bmod 599$, *όπως το υπολογίσαμε. Ας μην ξεχνάμε ότι παίρνουμε* ∞ *όταν επιχειρούμε να διαιρέσουμε με* 0, *οπότε εδώ ο λόγος που βρήκαμε τον παράγοντα* 599 *είναι ότι ο υπολογισμός του* $8!P$ *χρειάστηκε την διαίρεση με* $o \bmod 599$. □

3.3 Ανάλυση του Αλγόριθμου

Ας εξετάσουμε τη γενική περίπτωση. Θεωρούμε κατ' αρχάς μία ελλειπτική καμπύλη $\mathbb{E} \bmod p$, *για κάποιον πρώτο* p. *Ο μικρότερος θετικός ακέραιος* m *τέτοιος ώστε* $mP = \infty$ *πάνω από την* $\mathbb{E}$, *διαιρεί το πλήθος* N *των σημείων της* $\mathbb{E} \bmod p$, [3] *δηλαδή ισχύει και* $NP = \infty$. *Συχνά, το* m *θα είναι το ίδιο το* N *ή κάποιος μεγάλος διαιρέτης του* N. *Σε κάθε περίπτωση, αν* N *είναι γινόμενο μικρών πρώτων παραγόντων, τότε για κατάλληλη επιλογή λογικά μικρού φράγματος ομαλότητας* B, [4] *το* $B!$ *θα είναι πολλαπλάσιο του* N.

Από το θεώρημα του Hasse (θεώρημα 2.4.1), συμπεραίνουμε ότι ο N *είναι ακέραιος κοντά στο* p. *Αποδεικνύεται ότι η πυκνότητα ομαλών ακεραίων είναι αρκετά μεγάλη (αφήνουμε εδώ τις έννοιες μικρός και μεγάλος στην διαίσθηση, χωρίς αυστηρό ορισμό) ώστε αν επιλέξουμε τυχαία ελλειπτική καμπύλη* $\mathbb{E} \bmod p$ *τότε υπάρχει ικανοποιητικά μεγάλη πιθανότητα ότι η τάξη* N *της ομάδας των σημείων της* $\mathbb{E}$ *είναι ομαλός αριθμός. Πρακτικά, αυτό σημαίνει ότι αναμένεται ότι η μέθοδος*

[3] Από το γνωστό Πόρισμα του Θεωρήματος Lagrange.

[4] Βλ. ορισμό 3.1.1.

παραγοντοποίησης μέ χρήση ελλειπτικών καμπύλων αναμένεται με αρκετά καλή πιθανότητα να επιστρέψει τον p για αυτή την επιλογή καμπύλης. Αν δοκιμάσουμε παράλληλα περισσότερες της μίας ελλειπτικές καμπύλες $\mathbb{E} \pmod{n}$, όπου $n = pq$, τότε είναι πιθανό ότι για τουλάχιστον μία από τις καμπύλες $\mathbb{E} \pmod{p}$ ή $\mathbb{E} \pmod{q}$ η τάξη της ομάδας των σημείων της θα είναι ομαλός ακέραιος.

Συνοψίζοντας, όπως αναφέραμε και προηγούμενα, η ουσιαστική διαφορά της $p-1$ μεθόδου παραγοντοποίησης του Pollard με τον αλγόριθμο του Lenstra έγκειται στο γεγονός ότι δουλεύοντας πάνω από σώματα $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, έχουμε δυνατότητα να χρησιμοποιήσουμε πληθώρα ομάδων, και μπορούμε να ρεαλιστικά να ελπίζουμε ότι θα βρούμε μία ομάδα η τάξη της οποίας δεν διαιρείται από κάποιον μεγάλο πρώτο ή από δύναμη πρώτου.

Ένα πλεονέκτημα της μέθοδος των ελλειπτικών καμπύλων έναντι της μεθόδου του Pollard είναι ότι η δεύτερη απαιτεί ο $p-1$ να είναι ομαλός, δηλαδή να έχει μικρούς πρώτους παράγοντες. Αντίθετα, η μέθοδος των ελλειπτικών καμπύλων απαιτεί μόνο να υπάρχουν ομαλοί ακέραιοι αριθμοί κοντά στον p, έτσι ώστε κάποιος τυχαία επιλεγμένος ακέραιος κοντά στο p να έχει ικανοποιητικά μεγάλη πιθανότητα να είναι ομαλός. Κατά συνέπεια, η μέθοδος παραγοντοποίησης με χρήση ελλειπτικών καμπύλων επιτυγχάνει πολύ πιο συχνά από την $p-1$ μέθοδο παραγοντοποίησης του Pollard.

Η μέθοδός μας φαίνεται να είναι η πλέον κατάλληλη για χρήση προς παραγοντοποίηση ακεραίων μέτριου μεγέθους, δηλαδή αριθμούς περίπου 40 με 50 ψηφίων. Για μεγαλύτερους ακέραιους, η μέθοδος ελλειπτικών καμπύλων γίνεται αρκετά δύσχρηστη και οι αλγόριθμοι του τετραγωνικού και του αριθμητικού κόσκινου φαίνεται ότι είναι ουσιαστικά ανώτεροι.

3.4 Υλοποίησης της Μεθόδου Παραγοντοποίησης με Ελλειπτικές Καμπύλες με Maple

Στα ακόλουθα χρησιμοποιούμε τη συνάρτηση gop που είδαμε στην παράγραφο 2.5, με τη διαφορά ότι όταν ο υπολογισμός του αθροίσματος δύο σημείων αποτυγχάνει για τον λόγο ότι όταν η κλίση της ευθείας που διέρχεται από τα σημεία αυτά δεν ορίζεται στο σώμα στο οποίο εργαζόμαστε, τότε η gop επιστρέφει έναν μη τετριμμένο παράγοντα του n.

Ορίζουμε και δύο νέες συναρτήσεις, την SuccAdd και την SuccDoub. Και οι δύο αυτές συναρτήσεις υπολογίζουν το σημείο xP πάνω στην καμπύλη $\mathbb{E} \pmod{n}$ την οποία έχουμε επιλέξει, για κάποιον ακέραιο x. Η μεν πρώτη, η SuccAdd κάνει τον υπολογισμό γραμμικά ξεκινώντας με το άθροισμα $P + P$ και προσθέτοντας

κατ᾽ επανάληψη P ώσπου να υπολογιστεί το xP, δηλαδή οι πράξεις που κάνει είναι οι $P+P$, $P+2*P$, $P+3*P$, ..., $P+(x-1)*P$. Δεν θα τη χρησιμοποιήσουμε καθώς το εξαγόμενο θα είναι το ίδιο, εκτός κι αν μία από τις δύο συναντήσει συντομότερα το ∞. Εν γένει όμως, η SuccAdd είναι αρκετά πιο αργή από την SuccDoub, κάτι που θα φανεί αν ο αναγνώστης αφιερώσει λίγο χρόνο για να τις ελέγξει τρέχοντάς τις για κάποια συγκεκριμένη είσοδο στο Maple. Η δε SuccDoub υλοποιεί τον αλγόριθμο επαναλαμβανόμενου διπλασιασμού όπως αυτός δόθηκε στο Λήμμα 3.2.3.

Δίνουμε μία εφαρμογή πάνω στο παράδειγμα 3.2.4.

```
> restart;
> with(Bits);
```

[And, FirstNonzeroBit, GetBits, Iff, Implies, Join, Nand, Nor, Not, Or, Settings, Split, String, Xor]

We define a procedure L in order to verify whether that the points we are about to use, are points on a given elliptic curve E, or not. If this statement is true, we expect to find L=0.

```
> L:= proc(G,P)
>   local  ECeval;
>   ECeval := (P[2])^2 - ((P[1])^3 + G[1]* P[1] + G[2]) mod (G[3]);
>   return ECeval;
> end:
```

The following procedure computes the coefficients of the point C = A + B, over the elliptic curve E.

```
> gop := proc(G,A,B)
>   local   a,b,p,xA,yA,xB,yB,xC,yC,C,slope;
>   a := G[1];
>   b := G[2];
>   p := G[3];
>
> #   The case whee  at least one of A and B is the point at infinity
>
>   if A = [infinity,infinity] then
>       if L(G,B)<>0 then
>          print(B, `is not a point of the elliptic curve.`);
>          return `computation infeasible`;
>       else
>          return B;
>       fi;
>   fi;
>   if B = [infinity,infinity] then
>      if L(G,A)<>0 then
>        print(A, `is not a point of the elliptic curve.`);
>        return `computation infeasible`;
>       else
>        return A;
>       fi;
>   fi;
>
> xA := A[1];
> yA := A[2];
> xB := B[1];
> yB := B[2];
>
>
>   if L(G,A)<>0 then
>       print(A, `is not a point of the elliptic curve.`);
>       return `computation infeasible`;
>   fi;
>
>   if L(G,B)<>0 then
>        print(B, `is not a point of the elliptic curve.`);
>        return `computation infeasible`;
>   fi;
>
> # The case  where A = B and  yA is not equal to 0
>   if (xA - xB) mod p = 0 and (yA - yB) mod p = 0 and yA mod p
<> 0 then
>      if gcd(p,2*yA) = 1 then
>           slope := (3*xA^2 + a) / (2*yA) mod p;
>           xC := slope^2 - xA - xB mod p;
>           yC := -yA + slope*(xA - xC) mod p;
>           C := [xC,yC];
>           return C;
```

```
>       else
>            print(‘The modular inverse of the slope’s denominator‘,
>                  2*yA, ‘does not exist‘);
>            print(‘Thus, computing the gcd of the numbers‘, p ,
>                  2*yA,‘produces a factor of‘,p);
>            print(‘The factor is ‘, gcd(p,2*yA));
>            return NULL;
>       fi;
>
```

```
> # The case where either
> #    xA = xB and yA is not equal to yB or
```

```
> #      A = B and yA is not equal to 0 or
>
>   elif ((xA - xB) mod p = 0 and (yA - yB) mod p <>0)
>         or ((xA - xB) mod p = 0 and yA mod p =0) then
>      C:= [infinity, infinity];
>   else
>
```

```
> # The case where xA is not equal to xB
>
>   if gcd(p,xA-xB) = 1 then
>      slope := (yA - yB) / (xA - xB) mod p;
>      xC := slope^2 - xA - xB mod p;
>      yC := -yA + slope*(xA - xC) mod p;
>      C := [xC,yC];
>      return C;
>   else
```

```
>      print(‘The modular inverse of the slope’s denominator ‘,xA-xB,
>            ‘does not exist‘);
>      print(‘Thus, computing the gcd of the numbers‘, p,xA-xB,
>            ‘produces a factor of‘,p);
>      print(‘The factor is ‘, gcd(p,xA-xB));
>      return NULL;
>      fi;
>   fi;
> end:
>
```

The following procedure, SucAdd, implementates successive adding in order to compute x*P,
i.e. P+P, P+2*P, P+3*P, ..., P+ (x-1)*P

```
> SucAdd:= proc(G,P,x)
>   local i, Temp;
>
>   for i from 1 to x do
>      Temp[i]:=[infinity,infinity];
>   od:
>
>   Temp[1]:= P;
>   for i from 1 to x-1 do
>      Temp[i+1] := gop(G,P,Temp[i]);
>   od;
>
>   return Temp[x];
>
> end:
>
```

*The following procedure , i.e. SucDoub , implementates the successive doubling method, in order to compute x*P, over some given elliptic curve E mod n. We start by denoting as X the vector at wich we store the digits of the binary representation of x. Variable w symbolizes the number of operants of X.*

We also use two local variables (vectors as well), S and R, which we, at first, instantiate

```
> SucDoub := proc(G,P,x)
>   local  w, invX, S, R, k, i, X;
>   invX:= Split(x);
>   w:=nops(invX);
>
>   for i from 1 to w do
>     X[i]:= invX[w-i+1];
>     S[i]:=[infinity,infinity]:
>     R[i]:=[infinity,infinity]:
> od:
>   for k from 1 to w do
>     if X[k] = 1 then
>       R[k]:=gop(G,P,S[k]);
>     else R[k]:=S[k];
>     fi;
>     if k<w then
>       S[k+1]:= gop(G, R[k], R[k])
>     fi;
> od;
>
>   return R[w];
>
> end:
```

Vector G provides the feedback for the elliptic curve E we will

work with, i.e.E : y^2 = (x^3 + G[1]* x + G[2])) mod (G[3])

```
> G:= [5,-5,455839];
> P:=[1, 1];
> x:=7!;
```

$$G := [5, -5, 455839]$$
$$P := [1, 1]$$
$$x := 5040$$

```
> SucDoub(G, P,x);
```

$$[70028, 403526]$$

```
> SucDoub(G,%,8);
```

‘The modular inverse of the slope’s denominator ‘, 315074, ‘does not exist‘

‘Thus, computing the gcd of the numbers‘, 455839, 315074, ‘produces a factor of‘, 455839

‘The factor is ‘, 599

Κεφάλαιο 4

Παράρτημα

Ο Αλγόριθμος για το Τετραγωνικό Κόσκινο (The Quadratic Field Sieve Algorithm)

4.1 Περιγραφή του Αλγόριθμου

Ας υποθέσουμε ότι επιθυμούμε να παραγοντοποιήσουμε τον θετικό ακέραιο n. Περιγράφουμε σε τέσσερα βήματα τον αλγόριθμο για το Τετραγωνικό Κόσκινο.

Βήμα 1. *Επιλέγουμε ένα πεπερασμένο σύνολο «μικρών πρώτων» $\mathcal{F} = \{p_1, p_2, \cdots, p_k\}$, όπου οι p_i είναι πρώτοι για κάθε $i = 1, 2, \cdots, k \in \mathbb{N}$, το οποίο καλούμε* **βάση παραγοντοποίησης**.

Στη συνέχεια, για κάθε θετικό ακέραιο i με $1 \leq i \leq k$, θεωρούμε $t = \pm i$. Υπολογίζουμε τις τιμές $z_t = \lfloor \sqrt{n} \rfloor + t$ και κρατάμε εκείνες τις τιμές του t, για τις οποίες ο $y_t := z_t^2 - n$ είναι **$\mathbf{p_k}$ - ομαλός** *(* **$\mathbf{p_k}$ - smooth**), *δηλαδή παραγοντοποιείται πλήρως στο $\mathcal{F}$, έστω*

$$y_t \;=\; z_t^2 - n \;=\; \pm p_1^{e_{t1}}\, p_2^{e_{t2}} \cdots p_k^{e_{tk}} \;=\; \pm \prod_{i=1}^{k} p_i^{e_{t\,i}}\,. \tag{4.1}$$

Η αναζήτηση σταματάει όταν βρεθούν $k+2$ το πλήθος τέτοιες τιμές.

Πριν προχωρήσουμε, ας παρατηρήσουμε ότι, αν ο πρώτος $p_i \in \mathcal{S}$ εμφανίζεται σε μία, τουλάχιστον, σχέση της μορφής (4.1), δηλαδή αν $e_{ti} \neq 0$ για κάποιο t, τότε $z_t^2 \equiv n \pmod{p_i}$, άρα $\left(\frac{n}{p_i}\right) = +1$. Επομένως, όταν κατασκευάζουμε το σύνολο $\mathcal{S}$, έχει νόημα να περιλαμβάνουμε σε αυτό μόνο πρώτους p οι οποίοι είναι τετραγωνικά υπόλοιπα του n, δηλαδή $\left(\frac{n}{p}\right) = +1$.

Βήμα 2. *Για κάθε t από αυτά που βρήκαμε στο Βήμα 1, σχηματίζουμε το δυαδικό $k+1$ - διάστατο διάνυσμα, $\mathbf{v_t} = (v_{t\,0}, v_{t\,1}, v_{t\,2}, \cdots, v_{t\,k})$, όπου $v_{t\,i}$ είναι το ελάχιστο μη αρνητικό ισοϋπόλοιπο του $e_{t\,i}$ modulo 2 για κάθε i, με $1 \le i \le k$, ενώ $v_{t\,0} = 0$ εάν $y_t > 0$, και $v_{t\,0} = 1$ εάν $y_t < 0$.*[1]

Βήμα 3. *Επιλέγουμε ένα υποσύνολο $\mathcal{S}$ από τις τιμές t που βρήκαμε στο Βήμα 1, έτσι ώστε αν θεωρήσουμε τις αντίστοιχες σχέσεις της μορφής (4.1) και τις πολλαπλασιάσουμε κατά μέλη, να προκύψει μία σχέση, στο δεξιό μέλος της οποίας, όλοι οι εκθέτες $\sum_t e_{t\,0}$, $\sum_t e_{t\,1}, \cdots, \sum_t e_{t\,k}$ να είναι άρτιοι, έστω $2a_0$, $2a_1, \cdots, 2a_k$, αντίστοιχα. Εδώ εννοείται ότι ο δείκτης t διατρέχει ένα υποσύνολο (μη γνήσιο, εφόσον από τις $2k$ το πλήθος τιμές, έχουμε κρατήσει $k+2$) του $\{\pm 1, \pm 2, \cdots, \pm k\}$· αυτό το σύνολο είναι εκείνο που ονομάζουμε $\mathcal{S}$. Με άλλα λόγια, για κάθε i, $i = 0$, 1, 2, $\cdots, k$,*

$$\sum_{t \in \mathcal{S}} v_{i\,t} \;\equiv\; 0 \pmod 2 \,. \tag{4.2}$$

Βήμα 4. *Αν έχουμε επιτύχει τα ανωτέτω βήματα, έχουμε*

$$\begin{aligned} z^2 := \left(\prod_{t \in \mathcal{S}} z_t \right)^2 &\overset{(4.1)}{\equiv} \prod_{t \in \mathcal{S}} y_t \overset{(4.1)}{=} \prod_{p_i \in \mathcal{F}} p_i^{\sum_{t \in \mathcal{S}} e_{t\,i}} = \prod_{p_i \in \mathcal{F}} (p_i^{a_i})^2 \\ &= \left(\prod_{p_i \in \mathcal{F}} p_i^{a_i} \right)^2 := y^2 \pmod n \,. \end{aligned}$$

Επομένως, σε αυτή την περίπτωση, παίρνουμε $z^2 \equiv y^2 \pmod n$, έτσι που, $(z-y)(z+y) \equiv 0 \pmod n$ και εφόσον $z \not\equiv \pm y \pmod n$, ο $\mu\kappa\delta(z \pm y, n)$ δίνει έναν μη τετριμμένο παράγοντα του n.

Ολοκληρώνουμε εδώ την περιγραφή του Αλγόριθμου με την ακόλουθη παρατήρηση. Για κάθε δείκτη i, με $i = 0$, 1, $\cdots, k$, το $\sum_{t \in \mathcal{S}} e_{t\,i}$, μπορεί να γραφθεί και ως $\sum_{t \in \mathcal{S}} e_{t\,i} x_t$, όπου το x_t είναι 1 ή 0, ανάλογα με το αν έχει επιλεγεί ή όχι, η ισότητα της μορφής (4.1), η οποία αντιστοιχεί στο συγκεκριμένο $t \in \mathcal{S}$. Εν ολίγοις, συμπεραίνουμε ότι η κατάλληλη επιλογή σχέσεων της μορφής (4.1) ανάγεται στην εύρεση μη τετριμμένης λύσης του συστήματος ισοτιμιών

$$e_{1i} x_1 + e_{2i} x_2 + \cdots + e_{mi} x_m \;\equiv\; 0 \pmod 2 \qquad (i = 1, 2, \cdots, k) \,,$$

όπου $m = \#\mathcal{S}$. Δηλαδή, έχουμε να λύσουμε το σύστημα

$$\mathbf{A}\,\mathbf{x} \;\equiv\; \mathbf{o} \pmod 2 \,,$$

[1] Μπορούμε να εκτελέσουμε αυτό το βήμα, την παραγοντοποίηση του y_t δηλαδή, με συνεχείς δοκιμαστικές διαιρέσεις. Υπάρχει επίσης και μία διαδικασία κοσκινίσματος για να προσδιοριστεί η (4.1), όμως δε θα την αναφέρουμε στην παρούσα εργασία.

όπου $\mathbf{A}$ *είναι ο* $m \times (k+1)$ *πίνακας, του οποίου το στοιχείο στη θέση* (i,j) *είναι το* v_{ij}*, όπως ορίστηκε στο Βήμα2,* $\mathbf{x}$ *είναι το διάνυσμα στήλη των αγνώστων και* $\mathbf{o}$ *είναι το μηδενικό διάνυσμα διάστασης* m.[2]

4.2 Παραδείγματα - Εφαρμογές του Αλγόριθμου για το Τετραγωνικό Κόσκινο

4.2.1 Παραγοντοποίηση του $n = 34087$ με χρήση του Maple

```
> restart;
> with(numtheory):
> with(padic):
> with(linalg):
```

```
Warning, the protected name order has been redefined and unprotected

Warning, the protected names norm and trace have been redefined and
unprotected
```

n is the number we want to factor

```
> n:= 34087·
```

$$n := 34087$$

According to R. A. Mollin, the optimal value of number k, which denotes the number of primes to be included in the Factor Base, is - from knowledge about the distribution of smooth integers close to sqrt[2](n) - known to be one chosen to be approximately :

```
> k:= floor(evalf(exp(sqrt(ln(n)*ln(ln(n))))));
```

$$k := 140$$

SmoothBound is the constant Mollin refers to as "smoothness bound B"

```
> SmoothBound:=50;
```

$$SmoothBound := 50$$

FactorBase is the set we refer to as FACTOR BASE and denote by F

FactorBase is a maple-list, not a maple-set. We begin its construction by assigning FactorBasethe value [2]. Theithprimefunction returns the ith prime number, where the first prime number is 2. From the second prime (3) and while the ithprime(i) is less than the SmoothBound, we check if the ith prime is a quadratic residue modulo n, and if so, we include the ith prime in the

[2]Το σύστημα αυτό μπορεί να λυθεί με μεθόδους Γραμμικής Άλγεβρας, για παράδειγμα με κλασική απαλειφή Gauss. Μάλιστα, εδώ θα αναγνωρίσει κανείς ένα σύστημα στο σώμα $\mathbb{Z}_2$.

FactorBase. Command nops( )gives the number of opperants of a list, i.e. here it returns the cardinality of our FactorBase.

```
> FactorBase:=[2]:
> for i from 2 while(ithprime(i)<SmoothBound) do
> if jacobi(n,ithprime(i))=1 then
> FactorBase:=[op(FactorBase),ithprime(i)] fi
> od:
> FactorBase;nops(%);
```

[2, 3, 7, 11, 13, 17, 19, 23, 31, 37, 41, 43, 47]

13

For **t** *from floor(sqrt(n))+1 to floor(sqrt(n))+2*k will search which numbers y = z^2-n are factorizable over FactorBase.*
enum_y *is a temp integer variable, counting the acceptable y 's. The final value of enum_y is the total number of these values y , it is denoted by numb_y and must be larger the the cardinality of FactorBase.*
v *is a temp array, intitially void, where the i-th coordinate of the array v is the exponent of the i-th prime of the FactorBase in the prime decomposition of the current value of y. For each value of enum_y from 1 to numb_y, at the end of the loop, the vector expon[enum_y] is assigned the value v.*
test *is a temp integer variable, initially set to 1. Each time we find a prime factor p (p in FactorBase) of the current value of y, we multiply testby p raised to the p-adic order of y. So at the end of the nested for-loop, the proceeding if-statement checks whether this value of y is acceptable, i.e. if y is entirelly decomposed into of primes from FactorBase Array ZZ is such that its i-th coordinate is the value of z which corresponds to the i-th (acceptable) value of y.*

```
> enum_y:=0:
> ind:=array(1..nops(FactorBase)):
> for i from 1 to nops(FactorBase) do ind[i]:=0 od:
> # arhikopoiisi tou ind
> for z from floor(evalf(sqrt(n)))+1 to floor(evalf(sqrt(n)))+2*k
do
> v:=[]:
> y:=z^2-n:
> test:=1:
> for j from 1 to nops(FactorBase) do
> e:=ordp(y,FactorBase[j]):
> if e>0 then ind[j]:=ind[j]+1 fi:
> v:=[op(v),e]:
> test:=test*FactorBase[j]^e
> od:
```

```
> if abs(y/test)=1 then
> enum_y:=enum_y+1:
> expon[enum_y]:=v:
> print('enum_y=',enum_y,'z=',z,'y=',y,'v=',v):
> ZZ[enum_y]:=z:
> fi:
> od:
> numb_y:=enum_y;
```

$enum\_y =, 1, z =, 185, y =, 138, v =, [1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$

$enum\_y =, 2, z =, 187, y =, 882, v =, [1, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 3, z =, 189, y =, 1634, v =, [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]$

$enum\_y =, 4, z =, 191, y =, 2394, v =, [1, 2, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 5, z =, 193, y =, 3162, v =, [1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]$

$enum\_y =, 6, z =, 194, y =, 3549, v =, [0, 1, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 7, z =, 196, y =, 4329, v =, [0, 2, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]$

$enum\_y =, 8, z =, 198, y =, 5117, v =, [0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]$

$enum\_y =, 9, z =, 201, y =, 6314, v =, [1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0]$

$enum\_y =, 10, z =, 205, y =, 7938, v =, [1, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 11, z =, 206, y =, 8349, v =, [0, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0]$

$enum\_y =, 12, z =, 208, y =, 9177, v =, [0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]$

$enum\_y =, 13, z =, 209, y =, 9594, v =, [1, 2, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]$

$enum\_y =, 14, z =, 210, y =, 10013, v =, [0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0]$

$enum\_y =, 15, z =, 211, y =, 10434, v =, [1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1]$

$enum\_y =, 16, z =, 212, y =, 10857, v =, [0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]$

$enum\_y =, 17, z =, 215, y =, 12138, v =, [1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 18, z =, 227, y =, 17442, v =, [1, 3, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 19, z =, 229, y =, 18354, v =, [1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]$

$enum\_y =, 20, z =, 232, y =, 19737, v =, [0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]$

$enum\_y =, 21, z =, 233, y =, 20202, v =, [1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]$

$enum\_y =, 22, z =, 236, y =, 21609, v =, [0, 2, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 23, z =, 241, y =, 23994, v =, [1, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0]$

$enum\_y =, 24, z =, 248, y =, 27417, v =, [0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0]$

$enum\_y =, 25, z =, 250, y =, 28413, v =, [0, 2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0]$

$enum\_y =, 26, z =, 254, y =, 30429, v =, [0, 3, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$

$enum\_y =, 27, z =, 259, y =, 32994, v =, [1, 3, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]$

$enum\_y =, 28, z =, 261, y =, 34034, v =, [1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]$

$enum\_y =, 29, z =, 272, y =, 39897, v =, [0, 2, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0]$
$enum\_y =, 30, z =, 275, y =, 41538, v =, [1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0]$
$enum\_y =, 31, z =, 278, y =, 43197, v =, [0, 1, 1, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0]$
$enum\_y =, 32, z =, 283, y =, 46002, v =, [1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0]$
$enum\_y =, 33, z =, 284, y =, 46569, v =, [0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0]$
$enum\_y =, 34, z =, 285, y =, 47138, v =, [1, 0, 2, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]$
$enum\_y =, 35, z =, 286, y =, 47709, v =, [0, 4, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]$
$enum\_y =, 36, z =, 300, y =, 55913, v =, [0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0]$
$enum\_y =, 37, z =, 303, y =, 57722, v =, [1, 0, 2, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]$
$enum\_y =, 38, z =, 305, y =, 58938, v =, [1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1]$
$enum\_y =, 39, z =, 313, y =, 63882, v =, [1, 3, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]$
$enum\_y =, 40, z =, 317, y =, 66402, v =, [1, 2, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]$
$enum\_y =, 41, z =, 322, y =, 69597, v =, [0, 2, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0]$
$enum\_y =, 42, z =, 324, y =, 70889, v =, [0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0]$
$enum\_y =, 43, z =, 327, y =, 72842, v =, [1, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0]$
$enum\_y =, 44, z =, 334, y =, 77469, v =, [0, 1, 2, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]$
$enum\_y =, 45, z =, 344, y =, 84249, v =, [0, 2, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0]$
$enum\_y =, 46, z =, 352, y =, 89817, v =, [0, 1, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]$
$enum\_y =, 47, z =, 362, y =, 96957, v =, [0, 6, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$
$enum\_y =, 48, z =, 363, y =, 97682, v =, [1, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0]$
$enum\_y =, 49, z =, 365, y =, 99138, v =, [1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0]$
$enum\_y =, 50, z =, 367, y =, 100602, v =, [1, 7, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$
$enum\_y =, 51, z =, 379, y =, 109554, v =, [1, 1, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0]$
$enum\_y =, 52, z =, 399, y =, 125114, v =, [1, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 1]$
$enum\_y =, 53, z =, 400, y =, 125913, v =, [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2]$
$enum\_y =, 54, z =, 404, y =, 129129, v =, [0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0]$
$enum\_y =, 55, z =, 413, y =, 136482, v =, [1, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0]$
$enum\_y =, 56, z =, 415, y =, 138138, v =, [1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0]$
$enum\_y =, 57, z =, 436, y =, 156009, v =, [0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0]$
$enum\_y =, 58, z =, 438, y =, 157757, v =, [0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0]$
$enum\_y =, 59, z =, 443, y =, 162162, v =, [1, 4, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$
$enum\_y =, 60, z =, 447, y =, 165722, v =, [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1]$
$enum\_y =, 61, z =, 448, y =, 166617, v =, [0, 4, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0]$
$enum\_y =, 62, z =, 455, y =, 172938, v =, [1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0]$

$$numb\_y := 62$$

Checks if there exist primes in FactorBase, which has not been used in the prime factorization of the y's that have been found. If so, restart : go to the top of this page and redefine FactorBase without these primes.

```
> for j from 1 to nops(FactorBase) do
> if ind[j]=0 then print(‘useless ‘,j) fi od;
```

Constructs step by step a matrix M, whose columns are the v's calculated above. At each step checks whether the next column to be added (= next v) is linearly independant from the previous ones and only then it concatenates the column to M. Stops when it attains a matrix M with rank(M) = # FactorBase. We denote rank(M) by rnk. The vector vchoice_col shows which v's have been used in the construction of M, i.e. vchoice_col = [i,j,...] means that the first column of M is v[i] (corresponding to the i-th found y), the second column is v[j] (corresponding to the j-th found y) etc.

```
> M:=array(1..nops(FactorBase),1..1):
> for j from 1 to nops(FactorBase) do
> M[j,1]:=expon[1][j]
> od:
> rnk:=1:
> vchoice_col:=[1]:
> for i from 2 to numb_y do
> if rank(concat(M,expon[i]))=rnk+1 then
> M:=concat(M,expon[i]):
> rnk:=rnk+1:
> vchoice_col:=[op(vchoice_col),i]:
> #print(‘rnk=‘,rnk);
> fi:
> if rnk=nops(FactorBase) then break fi:
> od:
> print(‘rnk=‘,rnk);
> print(‘vchoice_col = ‘, vchoice_col);
> # choice_matr:=transpose(M);
```

$$rnk =, 13$$

$$vchoice\_col \ = \ , [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15]$$

Defines ZeroMatrix, which is the zero matrix of diamension 1 times nops(FactorBase)

```
> ZeroMatrix :=array(1..nops(FactorBase)):
> for i from 1 to nops(FactorBase) do ZeroMatrix[i]:=0 od:
```

Below, new_row means that M will be augmented with the column v[new_row] and the corresponding homogeneous system will be solved mod 2. The augmented matrix M_augm must have the same rank as M, otherwise give new_row a different

value.

```
> print(‘rank of the augmented M must be equal to rank of M‘):
> for new_row from (rnk+1) to numb_y do
> tempM_augm:= concat(M,expon[new_row]):
> if type(rank(tempM_augm)=rank(M),’equation’) = ‘true‘ then
> print(ok);
> M_augm := tempM_augm:
> break:
> fi:
> od:
```

$$\textit{rank of the augmented } M \textit{ must be equal to rank of } M$$

$$\textit{ok}$$

Solves the homogeneous system

```
> Linsolve(M_augm,ZeroMatrix) mod 2;
```

$$[\_t_{11} + \_t_{12},\ \_t_{10} + \_t_{11},\ 0,\ \_t_{14} + \_t_{12},\ \_t_{14},\ \_t_{14},\ 0,\ 0,\ 0,\ \_t_{10},\ \_t_{11},\ \_t_{12},\ 0,\ \_t_{14}]$$

The coordinates of the vector coef2 *show which congruences must be multiplied*

```
> for i from 1 to (rnk+1) do
> subs({_t[i]=1},%);
> od:
> coef1:=%;
> coef2 :=array(1..rnk+1):
> for i from 1 to rnk+1 do
> coef2[i]:= coef1[i] mod 2
> od:
> coef2=evalm(coef2);
```

$$\textit{coef1} := [2,\ 2,\ 0,\ 2,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1]$$

$$\textit{coef2} = [0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1]$$

The product of the right-hand sides of the congruences above, is a number, the exponents in the prime factorization of which are given by the vector ***C****.Of course, all the exponents are even and the primes appearing in the factorization are those in* ***FactorBase*** *. The square root mod n of the product is denoted by Y.*

```
> C:=scalarmul(row(transpose(M_augm),1),coef2[1]):
> for i from 2 to rnk+1 do
> TeMp1:= scalarmul(row(transpose(M_augm),i),coef2[i]):
> C:=matadd(C,TeMp1):
> od:
> C=evalm(C);
> Y:=1:
> for i from 1 to nops(FactorBase) do
> TeMp2:=Y*FactorBase[i]^(C[i]/2):
> Y:=mods(TeMp2,n)
> od:
> ’Y’=Y;
```

$$C = [2, 8, 4, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0]$$

$$Y = 144$$

We previously found various ***t****'s. Now we form the product* ***X*** *of those* ***t[i], i=1,2,...*** *for which* ***coef2[i]=1.***

```
> X:=1:
> for i from 1 to rnk do
> if coef2[i]=1 then
> TeMp3:= X*ZZ[vchoice_col[i]]:
> X:=mods(TeMp3,n)
> fi:
> od:
> if coef2[rnk+1]=1 then
> TeMp4:=X*ZZ[new_row]:
> X:=mods(TeMp4,n)
> fi:
> 'X'=X;
```

$$X = -7421$$

X^2 = Y^2 (mod n) *must be true.*

```
> X^2-Y^2 mod n;
```

$$0$$

```
> gcd(X+Y,n); gcd(X-Y,n);
```

$$383$$

$$89$$

The factors of n that have been found
Testing the result

```
> ifactor(n);
```

$$(89)\ (383)$$

Ευρετήριο

Βιβλιογραφία

[1] *Richard A. Mollin,* An Introduction to Cryptography*, : Library of Congress Cataloging - in - Publication Data, Chapman & Hall / CRC 2001*

[2] *H. W. Lenstra, Jr.,* Factoring Integers with Elliptic Curves*, : Annals of Mathematics, 126 (1987), 649 - 673*

[3] *Wade Trape, Lawrence C. Washington,* Introduction to Cryptography with Coding Theory*, : Prentice Hall / Upper Saddle River 2002*

[4] *Neal Koblitz,* A Course in Number Theory and Cryptography*, : 1994, 1987 Springer - Verlag New York, Inc.*

[5] *Νικόλαος Γ. Τζανάκης,* Θεμελιώδης Θεωρία Αριθμών, *http://server.math.uoc.gr/ tzanakis/Courses/NumberTheory/textbook_main.pdf*

[6] *Dale Husemöller,* Elliptic Curves*, : 1987 Springer - Verlag New York, Inc.*