ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Στοιχεία Θεωρίας Αριθμών και Εφαρμογές στην Κρυπτογραφία

9ο εξάμηνο, Ακαδημαϊκή περίοδος 2012-2013

Υλοποίηση κρυπτοσυστήματος και ψηφιακής υπογραφής με χρήση ελλειπτικών καμπυλών

Νίκος Γιανναράκης	03108054
Ζωή Παρασκευοπούλου	03108152

04/03/2013

Περιεχόμενα

1 Ελλειπτικές καμπύλες

1.1 Εισαγωγή

Πολλά συστήματα κρυπτογραφίας βασίζονται σε πράξεις πάνω σε κάποια αλγεβρική ομάδα. Μπορούμε να χρησιμοποιήσουμε μία ελλειπτική καμπύλη για να ορίσουμε μία ομάδα και έπειτα περιορίζοντας τα σημεία αυτής να ορίσουμε ένα σώμα. Θα δείξουμε αρχικά τις πράξεις που ορίζονται πάνω σε μία τέτοια ομάδα στο πεδίο των πραγματικών αριθμών και μετά στο $\mathcal{F}_p$.

1.2 Ελλειπτικές καμπύλες στο $\mathcal{R}$

Μία ελλειπτική καμπύλη στο $\mathcal{R}$ μπορεί να οριστεί ως ένα set σημείων (x,y) που ικανοποιούν μία εξίσωση ελλειπτικής καμπύλης της μορφής:

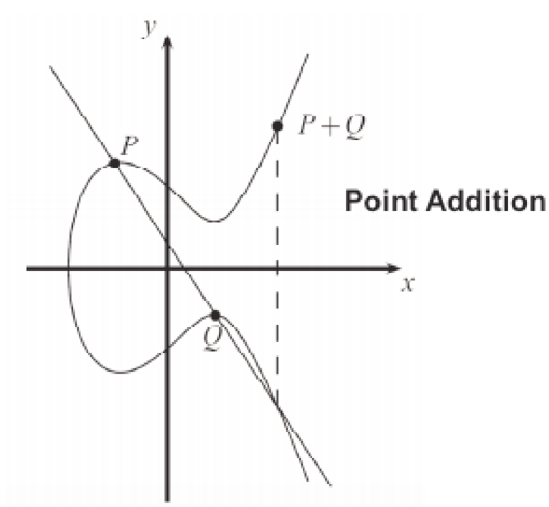
$$y^2 = x^3 + a \cdot x + b \ , \ x, y, a, b \in \mathcal{R}$$

Ανάλογα με την επιλογή των a και b έχουμε μία διαφορετική ελλειπτική καμπύλη. Για να ορίσουμε μία ομάδα από μία τέτοια ελλειπτική καμπύλη θα πρέπει το $x^3 + a \cdot x + b$ να μην έχει πολλαπλές ρίζες, δηλαδή να ισχύει $4 \cdot a^3 + 27 \cdot b^2 \neq 0$. Σε αυτή την περίπτωση μία τέτοια ομάδα ορίζεται απο τα σημεία που αποτελούν την ελλειπτική καμπύλη μαζί με ένα ακόμα σημείο $\mathcal{O}$ που θεωρείται το σημείο στο άπειρο.

1.2.1 Ορισμένες πράξεις σε ελλειπτικές καμπύλες πάνω στο $\mathcal{R}$

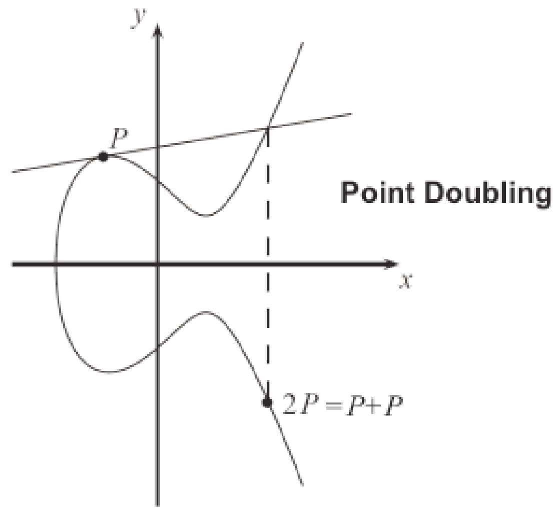
Αντίθετο σημείο Αν για δύο σημεία $P = (x_P, y_P)$ και $Q = (x_Q, y_Q)$ ισχύει ότι $Q = (x_P, -y_P)$ τότε λέμε ότι $P = -Q$. Γεωμετρικά αυτό σημαίνει οτι το Q είναι συμμετρικό του P ως προς τον άξονα x.

Πρόσθεση δύο σημείων Θα ορίσουμε την πρόσθεση δύο σημείων σε μία ελλειπτική καμπύλη γεωμετρικά. Έστω δύο σημεία P και Q για τα οποία ισχύει ότι $P \neq -Q$. Για να υπολογίσουμε το $R = P + Q$ φέρουμε μία ευθεία που τέμνει και τα δύο σημεία. Η ευθεία αυτή θα τέμνει την καμπύλη σε ακρίβως ένα σημείο ακόμα, το $-R$. Το αντίθετο αυτού είναι το άθροισμα $P + Q = R$. Για την περίπτωση όπου $P = -Q$ ισχύει $P + (-P) = \mathcal{O}$. Επίσης ισχύει ότι $P + \mathcal{O} = P$.



Σχήμα 1: Πρόσθεση δύο σημείων [4]

Διπλασιασμός σημείου Για την πρόσθεση ενός σημείου P στον εαυτό του φέρουμε ευθεία εφαπτόμενη στο P. Αν $y_P \neq 0$ τότε αυτή θα τέμνει την καμπύλη σε ένα ακόμα σημείο, έστω $-R$. Ισχύει ότι $P + P = 2 \cdot P = R$. Στην περίπτωση που $y_p = 0$ τότε $P + P = 2 \cdot P = \mathcal{O}$.



Σχήμα 2: Διπλασιασμός σημείου [4]

1.3 Ελλειπτικές καμπύλες πάνω από το σώμα $\mathbb{F}_p$.

Οι πράξεις πάνω σε πραγματικούς αριθμούς είναι αργές και στερούνται ακρίβειας λόγω στρογγυλοποιήσεων. Καθώς οι εφαρμογές κρυπτογραφίας απαιτούν ταχύτητα και ακρίβεια στις πράξεις προτιμούνται ελλειπτικές καμπύλες στο σώμα $\mathbb{F}_p$ ή F_{2^m}. Για να ορίσουμε μία ελλειπτική καμπύλη στο $\mathbb{F}_p$ αρκεί να διαλέξουμε $a, b \in \mathbb{F}_p$. Όλα τα σημεία (x, y) της καμπύλης θα ικανοποιούν την εξίσωση αυτής $modulo\, p$.

1.3.1 Ορισμένες πράξεις σε σε ελλειπτικές καμπύλες πάνω στο $\mathbb{F}_p$

Μία γεωμετρική προσέγγιση θα αποτύχει σε αυτή την περίπτωση λόγω του πεπερασμένου πλήθους σημείων. Για το λόγο αυτό θα χρησιμοποιήσουμε τις αντίστοιχες αλγεβρικές εξισώσεις.

Αντίθετο σημείο Αν για δύο σημεία $P = (x_P, y_P)$ και $Q = (x_Q, y_Q)$ ισχύει ότι $Q = (x_P, -y_P)$ τότε λέμε ότι $P = -Q$.

Πρόσθεση δύο σημείων Έστω δύο σημεία P και Q για τα οποία ισχύει ότι $P \neq -Q$. Έστω ο συντελέστης της ευθείας απο το P στο Q

$$s = \frac{(y_P - y_Q)}{(x_P - x_Q)} \pmod{p}$$

Για το $R = P + Q$ θα ισχύει:

$$\begin{aligned} x_R &= s^2 - x_P - x_Q \pmod{p} \\ y_R &= -y_P + s \cdot (x_P - x_R) \pmod{p} \end{aligned}$$

Διπλασιασμός σημείου Αν $y_P \neq 0$ τότε $P + P = 2 \cdot P = R$ όπου το R υπολογίζεται από τις παρακάτω σχέσεις:

$$\begin{aligned} s &= \frac{(3 \cdot x_P^2 + a)}{2 \cdot y_P} \pmod p \\ x_R &= s^2 - 2 \cdot x_P \pmod p \\ y_R &= -y_p + s \cdot (x_P - x_R) \pmod p \end{aligned}$$

1.4 Ελλειπτικές καμπύλες πάνω από το σώμα F_{2^m}.

Τα στοχεία του σώματος F_{2^m} είναι m-bit strings για το λόγο αυτό οι υπολογιστές μπορούν να εκτελέσουν αριθμητικές πράξεις πάνω σε αυτά πολύ αποδοτικά. Οι πράξεις δε διαφέρουν από αυτές που ορίσαμε παραπάνω.

2 Το πρόβλημα του διακριτού λογαρίθμου σε ελλειπτικές καμπύλες (ECDLP)

Κάθε σύστημα κρυπτογραφίας βασίζεται σε ένα υπολογιστικό πρόβλημα, συνήθως δύσκολο στον υπολογισμό του απουσία κάποιας πληροφορίας, π.χ. ενός secret key. Μπορούμε να φτιάξουμε συστήματα κρυπτογραφίας που βασίζονται στη δυσκολία υπολογισμού του διακριτού λογάριθμου σε ελλειπτικές καμπύλες (ECDLP).

2.1 Βαθμωτός Πολλαπλασιασμός

Ο βαθμωτός πολλαπλασιασμός ενός σημείου P της ελλειπτικής καμπύλης πάνω στο $\mathbb{F}_p$ με έναν ακέραιο k μικρότερο της τάξης του P ορίζεται ώς ένα νέο σημείο $R = k \cdot P = P + P + \cdots + P$ και μπορεί να επιτευχθεί με τις πράξεις πρόσθεσης και διπλασιασμού σημείου που ορίζονται στην ομάδα που ορίζει μια ελλειπτική καμπύλη στο $\mathbb{F}_p$

2.2 ECDLP

Με βάση λοιπόν τον βαθμωτό πολλαπλασιασμό ορίζουμε το διακριτό πρόβλημα του λογαρίθμου σε ελλειπτικές καμπύλες ως εξής:
Έστω $Q = k{\cdot}P$ όπου Q, P γνωστά σημεία της ελλειπτικής καμπύλης και k ένας ακέραιος. Το k ονομάζεται διακριτός λόγαριθμος του Q στη βάση P και ζητούμενο του προβλήματος είναι ο υπολογισμός του.

2.3 Elliptic curve Diffie-Hellman (ECDH)

Το σχήμα ανταλλαγής κλειδιού Diffie-Hellman για ελλειπτικές καμπύλες ακολουθεί την ίδια λογική με το Diffie-Hellman και στηρίζεται στο ECDLP. Η διαδικασία που ακολουθείται παρουσιάζεται παρακάτω:
Αρχικά επιλέγονται δημόσια ένα πεπερασμένο σώμα $\mathbb{F}_p$, μία ελλειπτική καμπύλη πάνω σε αυτό το σώμα και ένα σημείο G αυτής (domain parameters).
Έπειτα οι χρήστες A και B κάνουν τα παρακάτω:

- Επιλέγουν τυχαία έναν αριθμό d_A και d_B αντίστοιχα για τους οποίους ισχύει ότι $d_A < ord(G)$ και $d_B < ord(G)$.
- Υπολογίζουν το $Q_A = d_A \cdot G$ και $Q_B = d_B \cdot G$ με χρήση βαθμωτού πολλαπλασιασμού.
- Έχοντας σχηματίσει ένα ζεύγος public-private key (Q_A, d_A) και (Q_B, d_B) αντίστοιχα δημοσιεύουν τα Q_A, Q_B.
- Ο χρήστης A υπολογίζει το $K = d_A \cdot Q_B$ και ο χρήστης B το $K = d_B \cdot Q_A$.
- Έτσι τελικά και οι 2 έχουν υπολογίσει το $K = d_A \cdot d_B \cdot G = d_B \cdot d_A \cdot G$, χωρίς να είναι εφικτό για κάποιον τρίτο να το υπολογίσει χωρίς να λύσει το πρόβλημα του διακριτού λογαρίθμου για ελλειπτικές καμπύλες.

2.4 Elliptic curve digital signature algorithm (ECDSA)

Παραγωγή υπογραφής Για να υπογράψει ένα μήνυμα m, ο χρήστης A με παραμέτρους $D = (q, FR, a, b, G, n, h)$ και ένα ζεύγος ιδιωτικού-δημόσιου κλειδιού (d, Q) ακολουθεί τα παρακάτω βήματα:

1. Επιλέγει έναν τυχαίο αριθμό k τέτοιο ώστε $1 \leq k \geq n - 1$.
2. Υπολογίζει το σημείο $k \cdot G = (x_1, y_1)$.
3. Υπολογίζει το $r = x_1 \pmod n$. Αν $r = 0$ επιστρέφει στο βήμα 1.
4. Υπολογίζει το $k^{-1} \pmod n$.
5. Υπολογίζει το $SHA - 1(m)$ και μετατρέπει το αποτέλεσμα του bit-string σε έναν ακέραιο e.
6. Υπολογίζει το $s = k^{-1} \cdot (e + d \cdot r) \pmod n$. Αν $s = 0$ επιστρέφει στο βήμα 1.
7. Η υπογραφή του A για το μήνυμα m είναι (r, s).

Επαλήθευση υπογραφής Για να επαληθέυση μία υπογραφή (r, s) σε ένα μήνυμα m, ο χρήστης B με παραμέτρους $D = (q, FR, a, b, G, n, h)$ και ένα δημόσιο κλειδί Q θα πρέπει αρχικά να επαληθεύσει την ορθότητα των D και Q καθώς μπορεί το Q να έχει τροποποιηθεί απο κάποιον κακόβουλο χρήστη [1], έπειτα ακολουθεί τα παρακάτω βήματα:

1. Επιβεβαιώνει ότι τα r, s είναι ακέραιοι στο διάστημα $[1, n - 1]$.
2. Υπολογίζει το $SHA - 1(m)$ και μετατρέπει το αποτέλεσμα του bit-string σε έναν ακέραιο e.
3. Υπολογίζει το $w = s^{-1} \pmod n$.
4. Υπολογίζει το $u_1 = e \cdot w \pmod n$ και το $u_2 = r \cdot w \pmod n$.
5. Υπολογίζει το $X = u_1 \cdot G + u_2 \cdot G$
6. Εάν $X = \mathcal{O}$ τότε απορρίπτει την υπογραφή. Αλλιώς υπολογίζει το $u = x_1 \pmod n$ όπου x_1 η συντεταγμένη x του X.
7. Δέχεται την υπογραφή αν και μόνο αν $u = r$.

3 Υλοποίηση ενός συστήματος κρυπτογραφίας βασισμένο σε ελλειπτικές καμπύλες

3.1 Πλεονεκτήματα χρήσης

Ένας από τους κύριους λόγους χρησιμοποιήσης ελλειπτικών καμπυλών για υλοποίηση συστημάτων κρυπτογραφίας είναι ότι μπορούν να προσφέρουν τον ίδιο βαθμό ασφάλειας με συστήματα όπως το RSA ή το Diffie-Hellman με πολύ μικρότερο μήκος κλειδιού. Έτσι μειώνεται το υπολογιστικό κόστος χωρίς να επηρεάζεται ο βαθμός ασφάλειας. Στον παρακάτω πίνακα παρουσιάζεται το απαιτούμενο μήκος κλειδιού σε bits για το ECC ώστε να επιτευχθεί ανάλογος βαθμός ασφάλειας με αυτή των RSA και AES για διάφορα μήκη κλειδιών.

ECC	RSA	Αναλογία	AES
163	1024	1:6	
256	3072	1:12	128
384	7680	1:20	192
512	15360	1:30	256

Σχήμα 3: Αντιστοιχία μήκος κλεδιού σε bits του ECC με RSA και AES

Για το λόγο αυτό τα συστήμα κρυπτογραφίας βασισμένα σε ελλειπτικές καμπύλες χρησιμοποιούνται ευρέως σε συσκευές με περιορισμένες υπολογιστικές δυνατότητες και σε συσκευές που απαιτείται χαμηλή κατανάλωση ενέργειας, όπως κινητά για παράδειγμα.

3.2 Επιλογή παραμέτρων

Για την αποφυγή επιθέσεων προς το κρυπτοσύστημα απαιτείται κατάλληλη επιλογή των παραμέτρων αυτού. Οι παράμετροι αυτοί είναι:

Παράμετρος	Περιγραφή
p	Η χαρακτηριστική του πεπερασμένου σώματος $\mathbb{F}_p$
a	Ο συντελεστής a της ελλειπτικής καμπύλης
b	Ο συντελεστής b της ελλειπτικής καμπύλης
G	Ένα σημείο $G = (x_G, y_G)$ της ελλειπτικής καμπύλης (base point)
n	Η τάξη του στοιχείου G
h	$\sharp E(\mathbb{F}_p)/n$

Σχήμα 4: Παράμετροι ECC

Οι παραπάνω παράμετροι είναι πολύ σημαντικοί για την ασφάλεια του κρυπτοσυστήματος για αυτό και χρειάζεται ιδιαίτερη μέριμνα στην επιλογή τους. Σχετικά με το ποίες είναι οι επιθυμητές ιδιότητες ασφάλειας των παραμέτρων, πως να παράγουμε παραμέτρους με βάση αυτές και πως να επικυρώσουμε ότι ένα set παραμέτρων έχει τις ιδιότητες αυτές ο αναγνώστης προτρέπεται να ανατρέξει στα [3], [2] και[1]. Επίσης μπορούμε να χρησιμοποιήσουμε και ένα έτοιμο set παραμέτρων, τακτική που ακολουθήσαμε και εμείς στην υλοποίηση μας με το *brainpoolP256r1* [3].

3.3 Επίπεδα υλοποίησης

Η υλοποίηση ενός συστήματος κρυπτογραφίας μπορεί να γίνει σε 4 επίπεδα:

- Στο χαμηλότερο επίπεδο είναι η υλοποίηση modulo αριθμητικής που υπολογιστικά είναι και το πιο ακριβό μέρος.
- Υλοποίηση των ορισμένων πράξεων της ομάδας, δηλαδή της πρόσθεσης και του διπλασιασμού σημείου.
- Υλοποίηση του βαθμωτού πολλαπλασιασμού
- Στο υψηλότερο επίπεδο είναι η υλοποίηση ενός κρυπτοσυστήματος όπως το ECDH και το ECDSA.

3.4 Αλγόριθμοι υλοποίησης βαθμωτού πολλαπλασιασμού

Ο πιο απλός τρόπος να υπολογίσουμε το $k \cdot P$ είναι να κάνουμε k προσθέσεις του σημείου P με τον εαυτό του, ωστόσο αυτό δεν είναι καθόλου αποδοτικό. Για να κατασκευάσουμε ένα αποδοτικό κρυπτοσύστημα χρειαζόμαστε έναν αποδοτικό τρόπο εκτέλεσης βαθμωτού πολλαπλασιασμού. Ένας αποδοτικός τρόπος είναι με τη μέθοδο double-and-add [4].

Input: Elliptic curve E, elliptic curve point P, scalar d: $(d_1 d_2 \ldots d_t)$
Output: $T = d \cdot P$
$T \leftarrow P$
for $i \leftarrow t-1$ **downto** 0 **do**
    $T \leftarrow T + T \pmod{n}$
    if $d_i = 1$ **then**
        $T \leftarrow T + P \pmod{n}$
    end
end
return T

Algorithm 1: Μέθοδος double-and-add

Υπάρχουν διάφορες παραλλαγές αυτού του αλγορίθμου που στην πράξη είναι πιο αποδοτικοί [5].

4 Υλοποίηση στη γλώσσα OCaml

4.1 Υλοποίηση βιβλιοθήκης

Προκειμένου να υλοποιήσουμε ένα σύστημα κρυπτογραφίας βασισμένο σε ελλειπτικές καμπύλες υλοποιήσαμε μία βιβλιοθήκη που περιέχει όλες τις απαιτούμενες συναρτήσεις.

4.1.1 Modulo αριθμητική

Οι πράξεις modulo είναι το πιο ακριβό υπολογιστικά κομμάτι ενός συστήματος κρυπτογραφίας. Για το λόγο αυτό οι προσπάθειες βελτίωσης της απόδοσης του συστήματος πρέπει να επικεντρωθούν στο κομμάτι αυτό. Για τον παραπάνω λόγο για την υλοποίηση modulo αριθμητικής σε μεγάλους αριθμούς προτιμήσαμε τη βιβλιοθήκη μεγάλων αριθμών Zarith που βασίζεται στο GMP καθώς διαθέτει αρκετές και αποδοτικές συναρτήσεις λόγω του ότι το GMP είναι υλοποιημένο σε C και αρκετά optimized σε σχέση με την ενσωματωμένη βιβλιοθήκη της OCaml ή κάποια δική μας λύση.

4.1.2 Υλοποίηση πράξεων ομάδας

Έχοντας πλέον την υποστήριξη για modulo αριθμητική υλοποιήσαμε τις συναρτήσεις για τις πράξεις που ορίζονται σε μία ομάδα που ορίζει μία ελλειπτική καμπύλη όπως αυτές αναλύθηκαν στο κεφάλαιο 1.3.1.

4.1.3 Υλοποίηση βαθμωτού πολλαπλασιασμού

Χρησιμοποιήσαμε τη μέθοδο double-and-add όπως αναλύεται στο 3.4.

4.1.4 Ψηφιακή υπογραφή

Για τη διαδικασία υπογραφής και επαλήθευσης ενός μηνύματος ακολουθήσαμε τον αλγόριθμο που ορίζει το ECDSA με τη διαφορά ότι αντι για την προτεινόμενη secure hash function SHA-1 χρησιμοποιήσαμε το MD5.
Σημειώνεται ότι για πραγματικές εφαρμογές θα πρέπει να χρησιμοποιηθεί μία πιο ισχυρή συνάρτηση κατακερματισμού.

4.2 Προγράμματα επίδειξης

4.2.1 Υλοποίηση ECDH

Για να υλοποιήσουμε το σχήμα κρυπτογραφίας ECDH όπως αυτό αναλύθηκε στο 2.3 δημιουργήσαμε δύο προγράμματα που χρησιμοποιούν την παραπάνω βιβλιοθήκη, η λειτουργία των οποίων εξηγείται παρακάτω:

Register. Το πρόγραμμα *register_dh* δέχεται ως input ένα username και δημιουργεί ένα αρχείο με όνομα *username.pk* που περιέχει το δημόσιο κλειδί του χρήστη και ένα αρχείο με όνομα *username.sk* που περιέχει το ιδιωτικό κλειδί του χρήστη. Σε περίπτωση που υπάρχουν αρχεία με αυτό το όνομα δίνεται μήνυμα λάθους και το πρόγραμμα τερματίζει.
Σημείωνεται ότι το ιδιωτικό κλειδί του χρήστη αποθηκεύεται στο δίσκο ως plain-text, ενώ σε πραγματική εφαρμογή θα έπρεπε να κρυπτογραφείται.

Exchange. Το πρόγραμμα *exchange_dh* δέχεται ως input το όνομα ενός χρήστη και το όνομα του χρήστη με τον οποίο θέλει να ανταλλάξει κλειδί. Στη συνέχεια διαβάζει το δημόσιο κλειδί του δεύτερου χρήστη και το ιδιωτικό κλειδί του πρώτου χρήστη και υπολογίζει το κοινό κλειδί.

4.2.2 ECDSA

Για την υπογραφή μηνυμάτων δημιουργήσαμε δύο προγράμματα που χρησιμοποιούν τις συναρτήσεις *sign* και *verify* της βιβλιοθήκης η λειτουργία των οποίων εξηγείται παρακάτω:

Sign. Το πρόγραμμα *sign* δέχεται ως input το username ενός χρήστη και ένα μήνυμα και δημιουργεί ένα αρχείο με όνομα *username.msg* που περιέχει το μήνυμα και την υπογραφή. Σε περίπτωση που δε βρεθεί το username δίνεται μήνυμα λάθους και το πρόγραμμα τερματίζει.

Verify. Το πρόγραμμα *verify* δέχεται ως input το username του χρήστη στον οποίο ανήκει το μήνυμα που θέλουμε να επαληθεύσουμε, ανακτά το δημόσιο κλειδί του, βρίσκει το μήνυμα και την υπογραφή απο το αρχείο που έχει δημιουργήσει το *sign* και επαληθεύει την αυθεντικότητα της υπογραφής.

4.2.3 Πηγαίος κώδικας

Ο πηγαίος κώδικας για όλα τα παραπάνω βρίσκεται στο Github.

4.3 Τεκμηρίωση Βιβλιοθήκης

```
module Ecc :
  sig
```

```
    type point =
      | Infinity
      | Point of Z.t * Z.t
```

An elliptic curve point. It is either infinity or a point (x,y).

```
    type elliptic_curve = {
      p : Z.t ;
      a : Z.t ;
      b : Z.t ;
      g : point ;
      n : Z.t ;
      h : Z.t ;
    }
```

The type of domain parameters

```
    val inverse : Z.t -> Z.t -> Z.t
```

Ecc.inverse a n inverses the number a modulo n

```
    val verify_range : Z.t -> Z.t -> Z.t -> bool
```

Ecc.verify_range a l h returns true if $l \leq a \leq h$ or false otherwise

```
val is_point : point -> elliptic_curve -> bool
```

Returns true if a point belongs to an elliptic curve or false otherwise

```
val double_point : point -> elliptic_curve -> point
```

Given a point P on an elliptic curve and the elliptic curve retuns the point 2P on that curve.

```
val add_point : point -> point -> elliptic_curve -> point
```

Given two points P and Q, both on the same elliptic curve, and the elliptic curve returns P+Q on that curve.

```
val multiply_point : point -> Z.t -> elliptic_curve -> point
```

Given a point P on an elliptic curve, an integer k and the elliptic curve returns the scalar multiplication kP on that curve.

```
val integer_of_octStr : string -> Z.t
```

Converts an octet string to an intenger. Useful for defining domain parameters.

```
val brainpool_P256_r1 : elliptic_curve
```

An elliptic curve used for ECC as defined by Brainpool

```
val test_curve : elliptic_curve
```

An elliptic curve with small domain parameters for testing purposes

```
val random_big_int : Z.t -> Z.t
```

Ecc.random_big_int bound returns a random integer in `1, bound-1`

```
val sign : string -> Z.t -> elliptic_curve -> Z.t * Z.t
```

Ecc.sign message sk curve where sk is secret key of the user s and curve the public elliptic curve, returns the signature (r, s) of the message.

```
val verify : string -> Z.t * Z.t -> point -> elliptic_curve -> bool
```

Ecc.verify message (r, s) pk curve where pk is the public key of the user who signed the message, returns true if the (r, s) is a valid signature or false otherwise.

```
val create_keys : elliptic_curve -> point * Z.t
```

Creates a tuple (public_key, secret_key) where public_key is a point of the curve and secret_key an integer.

```
end
```

Αναφορές

[1] Don Johnson, Alfred Menezes, Scott Vanstone, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, *Certicom Research*.

[2] Certicom Research, *SEC 2: Recommended Elliptic Curve Domain Parameters*, September 2000.

[3] Brainpool, *ECC Brainpool Standard Curves and Curve Generation v1.0*, October 2005.

[4] Chrisoft Paar, Jan pelzl, *Understanding Cryptography: A textbook for Students and Practitioners*, *Springer*, July 2010.

[5] Daniel J. Bernstein, Tanja Lange, *Analysis and optimization of elliptic-curve single-scalar multiplication*.