

# Υλοποίηση ανταλλαγής κλειδιού DH και ψηφιακών υπογραφών βασισμένη σε ελλειπτικές καμπύλες

Νίκος Γιανναράκης  
Ζωή Παρασκευοπούλου

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Εθνικό Μετσόβιο Πολυτεχνείο

28 Ιανουαρίου 2013

# Κρυπτογραφία με ελλειπτικές καμπύλες

## Why elliptic curve cryptography?

- ▶ Αυξημένη ασφάλεια με μικρότερα μήκη κλειδιών
- ▶ Μειωμένο υπολογιστικό κόστος και bandwidth
- ▶ Ιδανικές για φορητές συσκευές λόγω ενεργειακών απαιτήσεων (κινητά κλπ.)
- ▶ ECC σε secure web servers, επιτάχυνση έως και 280% [6]

## Σύγκριση μήκους κλειδιού

ECC	RSA	Αναλογία	AES
160	1024	1:6	
256	3072	1:12	128
384	7680	1:20	192
512	15360	1:30	256

Σχήμα : Σύγκριση μήκους κλειδιού σε bits

	ECC-160	RSA-1024	ECC-224	RSA-2048
Time(ms)	3.69	8.75	5.12	56.18
Ops/Sec	271.3	114.3	195.5	17.8
Perf ratio	2.4 : 1.0		11.0 : 1.0	
Key ratio	1.0 : 6.4		1.0 : 9.1	

Σχήμα : Σύγκριση απόδοσης ανάλογα με το μήκος κλειδιού [6]

# Ελλειπτικές καμπύλες στο $\mathcal{R}$

## Ορισμός

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

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

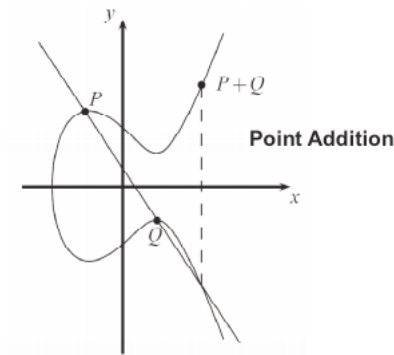
μαζί με ένα σημείο  $\mathcal{O}$ , το οποίο ονομάζουμε σημείο στο άπειρο.

## Ορισμός πράξεων

- ▶ Πρόσθεση δύο σημείων  $P, Q$
- ▶ Διπλασιασμός ενός σημείου  $P$

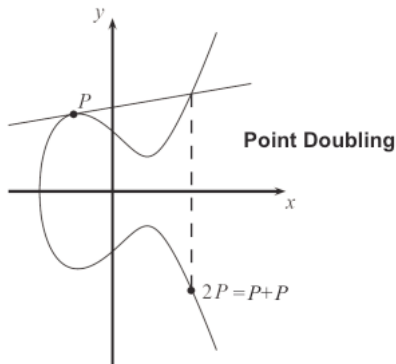
## Πρόσθεση δύο σημείων πάνω σε ελλειπτικές καμπύλες στο $\mathcal{R}$

Η πρόσθεση δύο σημείων  $P, Q$  μπορεί να οριστεί γεωμετρικά



## Διπλασιασμός σημείου πάνω σε ελλειπτικές καμπύλες στο $\mathcal{R}$

Ο διπλασιασμός ενός σημείου πάνω σε μία ελλειπτική καμπύλη ορίζεται γεωμετρικά σύμφωνα με το παρακάτω σχήμα



## Προβλήματα

- ▶ Αργές πράξεις σε πραγματικούς αριθμούς
- ▶ Έλλειψη ακρίβειας

# Ελλειπτικές καμπύλες πάνω από το $\mathbb{F}_p$ και το $\mathbb{F}_{2^m}$

## Ορισμός

Διαλέγοντας  $a, b \in \mathbb{F}_p$  και υπολογίζοντας τα σημεία  $(x, y)$  της καμπύλης *modulo*  $p$  ορίζουμε μία ελλειπτική καμπύλη στο  $\mathbb{F}_p$ .



## Πρόσθεση δύο σημείων πάνω σε ελλειπτικές καμπύλες στο $\mathbb{F}_p$

Η πρόσθεση δύο σημείων  $R = P + Q$  σε μία ελλειπτική καμπύλη στο  $\mathbb{F}_p$  ορίζεται αλγεβρικά:

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

$$x_R = s^2 - x_P - x_Q \pmod{p}$$

$$y_R = -y_P + s \cdot (x_P - x_R) \pmod{p}$$

Το  $\mathcal{O}$  είναι το ουδέτερο στοιχείο της πρόσθεσης :  $P + \mathcal{O} = P$ .

## Διπλασιασμός σημείου πάνω σε ελλειπτικές καμπύλες στο $\mathbb{F}_p$

Ο διπλασιασμός σημείου  $R = 2P$  σε μία ελλειπτική καμπύλη στο  $\mathbb{F}_p$  ορίζεται αλγεβρικά:

$$s = \frac{(3 \cdot x_P^2 + a)}{2 \cdot y_P} \pmod{p}$$

$$x_R = s^2 - 2 \cdot x_P \pmod{p}$$

$$y_R = -y_P + s \cdot (x_P - x_R) \pmod{p}$$

## Βαθμωτός πολλαπλασιασμός πάνω σε ελλειπτικές καμπύλες στο $\mathbb{F}_p$

Με χρήση των παραπάνω πράξεων μπορούμε να ορίσουμε την πράξη του βαθμωτού πολλαπλασιασμού  $R = k \cdot P$  όπου  $k \in \mathbb{Z}$  και  $P$  ένα σημείο ελλειπτικής καμπύλης.  $P = \mathcal{O} \rightarrow k \cdot P = \mathcal{O}$

- ▶ Naive  $P + P \dots + P$
- ▶ Double-and-add (το ανάλογο του επαναλαμβανόμενου τετραγωνισμού)
- ▶ Windowed, Sliding-window, wNAF, Montgomery ladder ... [3]

## Double-and-add

**Input:** Elliptic curve  $E$ , elliptic curve point  $P$ , scalar  $d$ :

$(d_0 d_1 \dots d_{t-1})$

**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

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

## Ορισμός

Έστω ελλειπτική καμπύλη στο  $\mathbb{F}_p$  και έστω δύο σημεία αυτής  $P, Q$ . Αν η τάξη του  $P$  είναι  $n$  τότε το πρόβλημα διακριτού λογαρίθμου ορίζεται ως η εύρεση ενός ακεραίου  $0 \leq l \leq n - 1$  τέτοιου ώστε  $Q = l \cdot P$ .

## Ανταλλαγή κλειδιού με τη μέθοδο Diffie-Hellman για ελλειπτικές καμπύλες (ECDH)

- ▶ Ο χρήστης A και ο χρήστης B επιλέγουν δημόσια τις παραμέτρους  $D = (q, a, b, G, n, h)$
- ▶ Ο χρήστης A επιλέγει έναν τυχαίο αριθμό  $1 \leq a \leq n - 1$  ως ιδιωτικό κλειδί και υπολογίζει και στέλνει στον B το δημόσιο κλειδί του  $a \cdot G$ .
- ▶ Ο χρήστης B επιλέγει έναν τυχαίο αριθμό  $1 \leq b \leq n - 1$  ως ιδιωτικό κλειδί και υπολογίζει και στέλνει στον A το δημόσιο κλειδί  $b \cdot G$  και το στέλνει στον A.
- ▶ Ο A υπολογίζει το  $a \cdot b \cdot G$
- ▶ Ο B υπολογίζει το  $b \cdot a \cdot G$
- ▶ Το κοινό κλειδί τους είναι το  $a \cdot b \cdot G = b \cdot a \cdot G$

# Ψηφιακές υπογραφές με τον αλγόριθμο DSA για ελλειπτικές καμπύλες (ECDSA)

## Παραγωγή υπογραφής

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

# Ψηφιακές υπογραφές με τον αλγόριθμο DSA για ελλειπτικές καμπύλες (ECDSA)

## Παραγωγή υπογραφής

1. Επιλέγει έναν τυχαίο αριθμό  $k$  τέτοιο ώστε  $1 \leq k \leq n - 1$ .
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  $(r, s)$ .



# Ψηφιακές υπογραφές με τον αλγόριθμο DSA για ελλειπτικές καμπύλες (ECDSA)

## Επαλήθευση υπογραφής

Για να επαληθεύσει μία υπογραφή  $(r, s)$  σε ένα μήνυμα  $m$ , ο χρήστης  $B$  παίρνει τις παραμέτρους  $D = (q, FR, a, b, G, n, h)$  και το δημόσιο κλειδί  $Q$  του  $A$  και ακολουθεί τα παρακάτω βήματα:

Προσοχή! Ο  $B$  θα πρέπει να ελέγξει ότι τα στοιχεία του  $A$  δεν έχουν αλλοιωθεί, π.χ. αν το σημείο  $Q$  ανήκει στην καμπύλη που ορίζεται από το  $D$

# Ψηφιακές υπογραφές με τον αλγόριθμο DSA για ελλειπτικές καμπύλες (ECDSA)

## Επαλήθευση υπογραφής

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  $u = r$ .

# Υλοποίηση

## Επιλογή παραμέτρων

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 $G = (x_G, y_G)$
$n$	Η τάξη του στοιχείου $G$
$h$	$\#E(\mathbb{F}_p)/n$

Σχήμα : Domain Parameters

Απαιτείται πολύ προσεκτική επιλογή των παραμέτρων [1] [2] [4]

# Υλοποίηση

## Επίπεδα υλοποίησης

- ▶ Αριθμητική modulo με υποστήριξη για μεγάλους αριθμούς
- ▶ Υλοποίηση των πράξεων που ορίζονται στην ομάδα (πρόσθεση, διπλασιασμός)
- ▶ Υλοποίηση του βαθμωτού πολλαπλασιασμού
- ▶ Υλοποίηση ενός κρυπτοσυστήματος π.χ. ECDH

# Υλοποίηση στη γλώσσα OCaml

## Why OCaml?

- ▶ **Μείωση σφαλμάτων.** Έλεγχος τύπων κατά τη μεταγλώττιση.
- ▶ **Βιβλιοθήκες.** Αρκετές έτοιμες συναρτήσεις (για big numbers κ.α.)
- ▶ **Ταχύτητα.** Απόδοση αρκετά κοντά σε low-level γλώσσες όπως C.
- ▶ **Garbage collected.** Η διαχείριση μνήμης γίνεται αυτόματα, ένα πράγμα λιγότερο για να ανησυχούμε.



# Υλοποίηση στη γλώσσα OCaml

## Βασικοί τύποι

```
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

# Υλοποίηση στη γλώσσα OCaml

## Modulo αριθμητική

Χρησιμοποιήθηκε η βιβλιοθήκη μεγάλων αριθμών Zarith, χρησιμοποιεί το GMP (C/C++). Διάφορες ετοιμες συναρτήσεις για βασικές πράξεις όπως εύρεση αντιστρόφου ενός αριθμού *modulo*  $n$ .

# Υλοποίηση στη γλώσσα OCaml

## Υλοποίηση πράξεων ομάδας

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

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



# Υλοποίηση στη γλώσσα OCaml

## Υλοποίηση βαθμωτού πολλαπλασιασμού

```
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.

# Υλοποίηση στη γλώσσα OCaml

## Υλοποίηση δημιουργίας ζεύγους κλειδιών

```
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.

# Υλοποίηση στη γλώσσα OCaml

## Υλοποίηση ψηφιακής υπογραφής

```
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.

# Βελτιστοποιήσεις

## Βελτιστοποίηση αριθμητικής modulo

Το πιο χρονοβόρο κομμάτι είναι οι πράξεις με μεγάλους αριθμούς και η αριθμητική modulo. Η χρήση της βιβλιοθήκης Zarith μας λύνει το πρόβλημα βελτιστοποιήσεις αυτού του κομματιού καθώς χρησιμοποιεί το GMP που έχει γρήγορες υλοποιήσεις πράξεων στη γλώσσα C.

## Βελτιστοποίηση πράξεων

Οι συναρτήσεις `add_point` και `double_point` δεν παρουσιάζουν ιδιαίτερα περιθώρια βελτιστοποίησης. Μπορούμε να βελτιστοποιήσουμε τη συνάρτηση `multiply_point` χρησιμοποιώντας έναν καλύτερο αλγόριθμο από αυτούς που έχουν αναφερθεί (η υλοποίηση μας χρησιμοποιεί τη μέθοδο `double-and-add`)

# Επιθέσεις

- ▶ Επίλυση ECDLP (πρακτικά ανέφικτο με κατάλληλη επιλογή παραμέτρων)
- ▶ Επίθεση στη συνάρτηση κατακερματισμού (hash function) εαν χρησιμοποιείται.
- ▶ Κακή διαχείριση κλειδιών (private key, αριθμός  $k$  στο ECDSA) κλπ.

# Επίλυση του ECDLP

## Certicom ECC Challenge

- ▶ Στόχος είναι να βρεθούν τα ιδιωτικά κλειδιά απο μία δοθείσα λίστα με δημόσια κλειδιά και τις αντίστοιχες παραμέτρους.
- ▶ Επίπεδο 1 : 109-bit, 131-bit
- ▶ Επίπεδο 2 : 163-bit, 191-bit, 239-bit, 359-bit
- ▶ Το 2004 λύθηκε το πρόβλημα για τα 109-bit από 10000 υπολογιστές σε 549 μέρες χρησιμοποιώντας τη μέθοδο rho. Για τα 131-bit όμως θα χρειαστούν σημαντικά περισσότεροι πόροι.
- ▶ Τα προβλήματα στο επίπεδο 2 θεωρούνται υπολογιστικά ανέφτικα.

# Επίθεση στο ECDSA

## Sony Playstation 3

Το playstation 3 χρησιμοποιεί το σχήμα ψηφιακής υπογραφής ECDSA για ψηφιακές υπογραφές στα παιχνίδια και στις αναβαθμίσεις του firmware του έτσι ώστε να μην επιτρέπεται σε unsigned κώδικα να εκτελεστεί στην κονσόλα.

# Επίθεση στο ECDSA

## Επιλογή τυχαίου αριθμού $k$

- ▶ Ο τυχαίος αριθμός  $k$  έχει τις ίδιες απαιτήσεις ασφάλειας με το ιδιωτικό κλειδί  $d$ .
- ▶ Αυτό συνεπάγεται απο το γεγονός οτι αν ο κακόβουλος χρήστης  $E$  ανακτήσει ένα  $k$  που χρησιμοποιεί ο  $A$  για να υπογράψει ένα μήνυμα  $m$  τότε μπορεί να ανακτήσει το ιδιωτικό κλειδί του  $A$  αφού  $d = r^{-1} \cdot (k \cdot s - e) \pmod{n}$ .
- ▶ Συνεπώς το  $k$  θα πρέπει να παράγεται με ασφαλή τρόπο και να αποθηκεύεται με ασφαλή τρόπο.



# Επίθεση στο ECDSA

## Sony Playstation 3 τρόπος επιλογής τυχαίου $k$

- ▶ Η Sony δε παρήγαγε ποτέ τυχαίο  $k$  αλλά χρησιμοποιούσε μία σταθερά για  $k$
- ▶ Με τον τρόπο που δείξαμε παραπάνω υπολογίστηκε το private key της Sony και δόθηκε η δυνατότητα να κάνουμε sign ότι κώδικα θέλουμε. [7]

## Παραγωγή τυχαίων αριθμών!

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

# References



[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  
*Certicom Research*



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



[4] Brainpool  
ECC Brainpool Standard Curves and Curve Generation v1.0  
*Brainpool*

# References



[5] Chrisof Paar, Jan pelzl,

Understanding Cryptography: A textbook for Students and Practitioners

*Springer*



[6] Vipul Gupta, Douglas Stebila, Stephen Fung, Sheueling Chang, Nils Gura, Hans Eberle

Speeding up secure web transactions using elliptic curve cryptography

*Sun Microsystems Labs*



[7] bushing, marcan, sgher, sven

PS3 Epic Fail

*failOverflow*

Fork here!

<https://github.com/zoep/ECC-OCaml>

The end