ELLIPTIC CURVE CRYPTOGRAPHY

A Java Implementation

WHY ELLIPTIC CURVE CRYPTOGRAPHY?

ECDLP

$P = d \cdot G$

The most efficient known algorithm to compute the secret d *is Pollard's* ρ *having a time complexity of*

$$O\left(\sqrt{\frac{\pi n}{2}}\right)$$

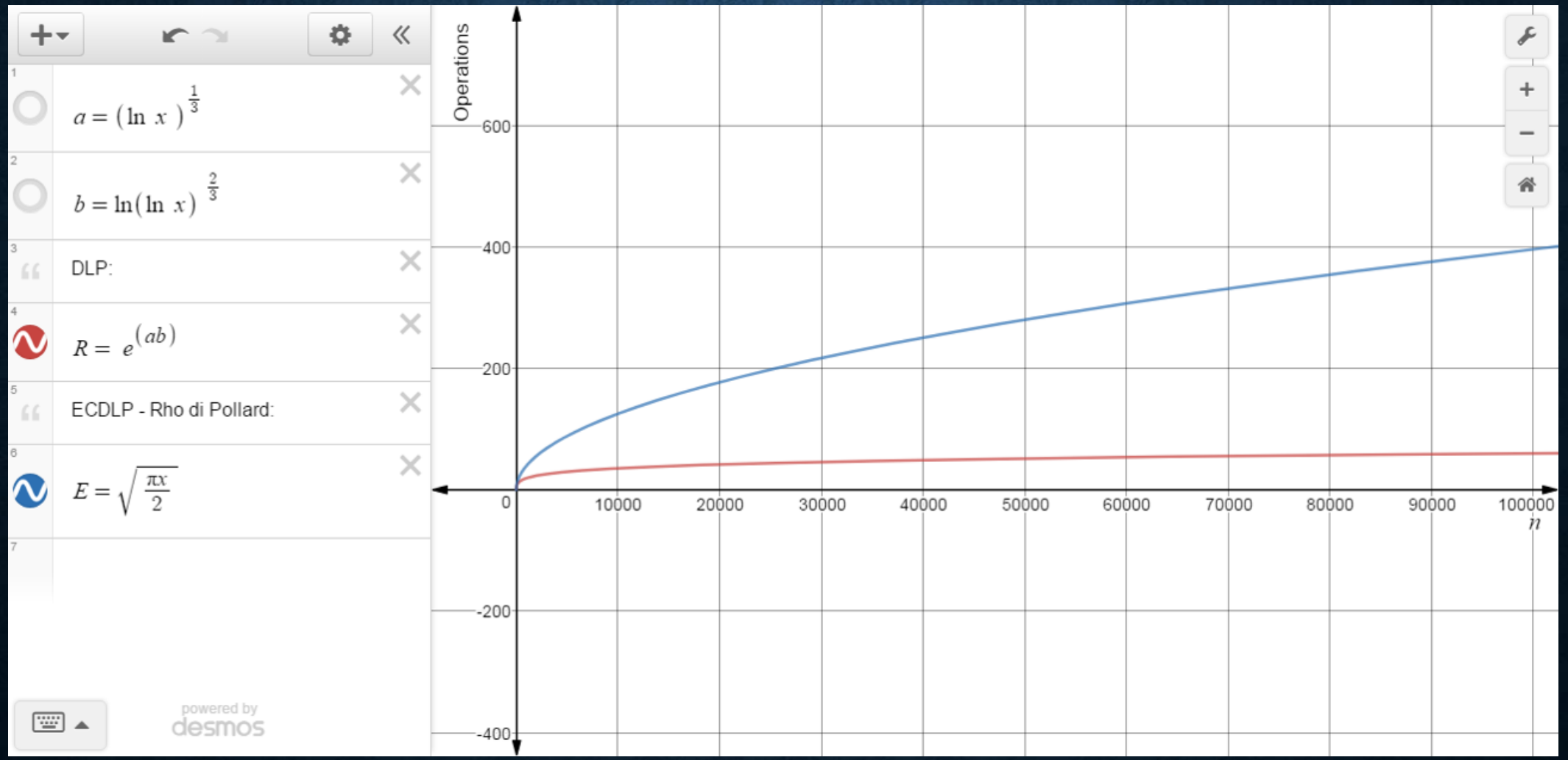
DLP

$$y = a^x \bmod(n)$$

The fastest know algorithm to find the x *requires a time complexity of*

$$O\left(e^{\sqrt[3]{\ln(n)\cdot\ln(\ln(n))}}\right)$$

DISCRETE LOGARITHM PROBLEM



AND WHAT ABOUT SECURITY?

Security Level	RSA Public Key	ECC Public Key	ECC / RSA
80	1024	160	15%
112	2048	224	10%
128	3072	256	8%
192	7680	384	5%
256	15360	512	3%

JAVA: WHAT HAS BEEN IMPLEMENTED

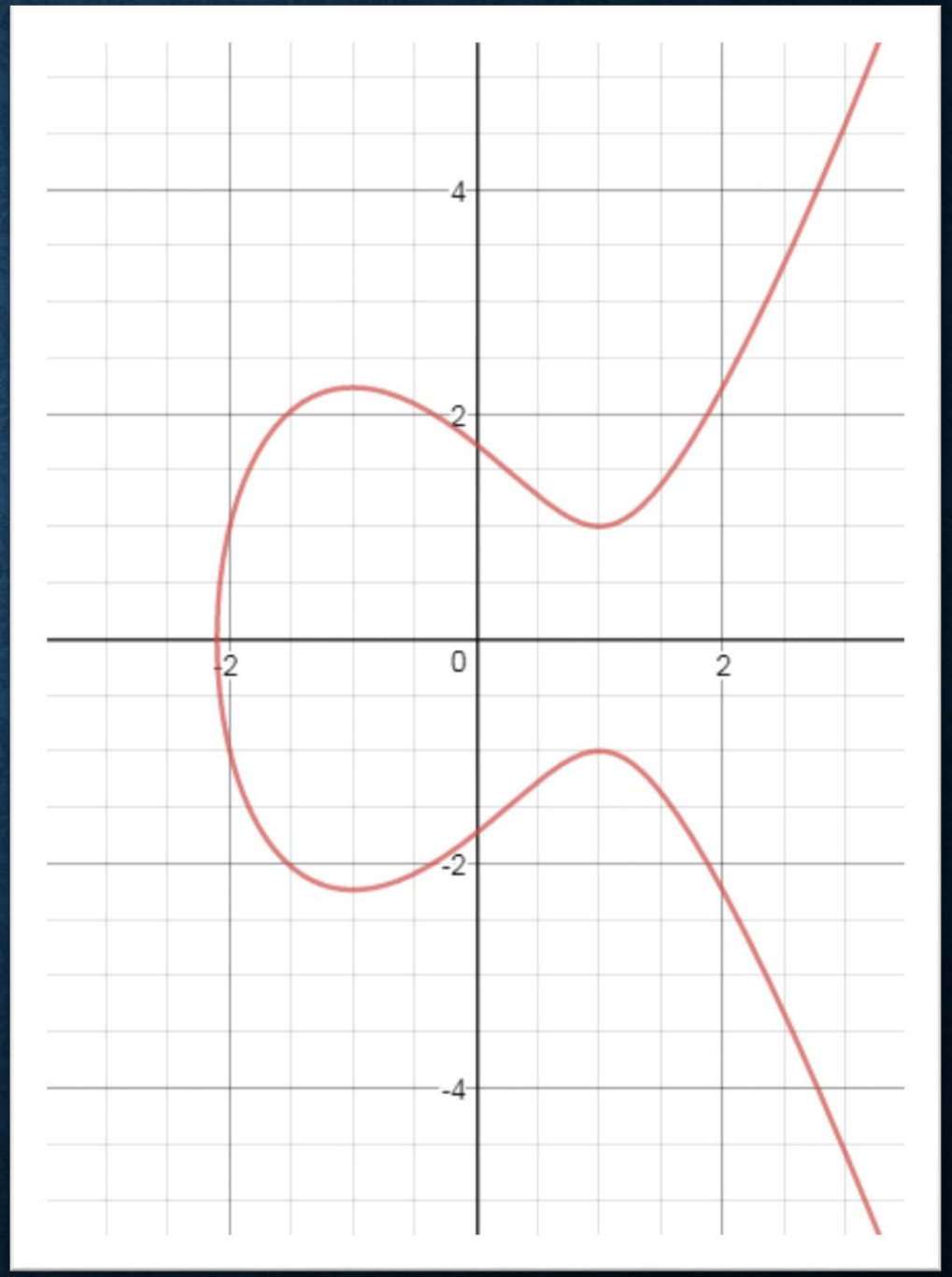
- Base class «*Curve*»
  - Necessary parameters
  - Operations: Addition, Doubling, Multiplication
  - Checks: Smoothness, Belonging…
  - Standardised, Verifibly Random, Custom curves

- Cryptographic Protocols

$$y^2 = x^3 - 3x + 3$$

EC necessary parameters:

- Modular Field in which it is immersed,
- Coefficients (-3 and 3 according to above)
- Generator Point G
- Order of generator
- Index/Cofactor
- Security Level



BASIC OPERATIONS

ADDITION:

Given $A = (x_A, y_A)$, $B = (x_B, y_B)$ their sum is $C = (x_C, y_C)$ as follows:

$$\begin{cases} m = \dfrac{y_A - y_B}{x_A - x_B} \\ x_C = m^2 - (x_A + x_B) \\ y_C = m(x_A - x_C) - y_A \end{cases}$$

DOUBLING:

Given the curve $\boldsymbol{y^2 = x^3 + ax + b}$ and one of its points $A = (x_A, y_A)$, one may compute $C = 2A = (x_C, y_C)$ via:

$$\begin{cases} m = \dfrac{3x_A^2 + \boldsymbol{a}}{2y_A} \\ x_C = m^2 - 2x_A \\ y_C = m(x_A - x_C) - y_A \end{cases}$$

SCALAR MULTIPLICATION

MULTIPLICATION (by a scalar):

Given a curve's point **A** and a scalar **k** we want to compute **C = k · A**

In order to ensure a high computational speed and a Side Channel Attacks immunity it has been chosen to implement the Montgomery Ladder algorithm.

In the pseudocode we assume to have initialised the following parameters:

$$\mathbf{k} \text{ base } 2 = d_j d_{j-1} \cdots d_2 d_1 d_0$$

$$P_1 = \boldsymbol{A}, \quad P_2 = 2\boldsymbol{A}$$

Algorithm 1 Montgomery Ladder

for $i = j - 1$ to 0 **do**
  if $d_i = 1$ **then**
    $P_1 = P_1 + P_2$
    $P_2 = 2P_2$
  else
    $P_2 = P_1 + P_2$
    $P_1 = 2P_1$
  end if
end for
return P_1

CHECKS AND OTHER OPERATIONS

- Compute Hasse's interval for curve's cardinality
- Check the curve is smooth: $4a^3 \neq 27b^2$
- Check a point belongs to a curve
- Check a curve is not Anomalous
- Check a curve is MOV resistant
- Compute curve's Security Level based on its modular field
- Check the Modular Field is Prime
- Check the index/cofactor is small enough

IMPLEMENTED CURVES

- Standardised: those curves whose parameters had been defined by governative agencies like NIST, Certicom, Brainpool etc. Most famous curves are: Nist's P192, Brainpool's brainpoolP160r1 and the Certicom's secp160r1.
- Verifiably Random: curves generated from a string seed. Via hashing operations and their concatenations one may compute a number «n» which bring to calculate the curve's coeffients «a», «b». The algorithm follows the ANSI X9.62 (1998), appendix "A.3.3.2 Elliptic curves over Fp".
- Custom: one may simply insert its parameters to create an EC. Upon instantiation many checks are needed to ensure parameters correctness and curve's cryptographic strength.

JAVA: WHAT HAS BEEN IMPLEMENTED

- Base class «*Curve*»
- Cryptographic Protocols
  - ECDH
  - ECDSA
  - ECIES / ECAES
  - ECMQV

CRYPTOGRAPHIC PROTOCOLS

- ECDH: Elliptic Curve Diffie-Hellman. Enables symmetric key agreement between two parties
- ECDSA: Elliptic Curve Digital Signature Algorithm. It's an EC variation of DSA for Digital Signature
- ECIES: Elliptic Curve Integrated Encryption Standard. Used to encrypt and decrypt messages. It is a Block Cypher with hashes
- ECAES: Elliptic Curve Augmented Encryption Standard. Improved ECIES version with a redundacy element and a MAC (*Message Authentication Code*) function
- ECMQV: Elliptic Curve Menezes Qu Vanston. Improvement of ECDH ensuring: greater security, MitM immunity, Forward Secrecy, improved speed and algorithm flexibility

Curve generated from the seed: "One must acknowledge with cryptography no amount of violence will ever solve a math problem."

Curve's equation: y^2 = x^3 + ax + b mod(p)
a =
73786917767457283531353903875153871174800986390846405399072403710997
38858521698648340789796619739569871216543021087020838210840240445054
62887801800605087153155502339191346165512430620116086197691756282639
10889440556397873269527505542151344588992194802508034868063486617851
732353017537218467382336381147350936
b =
73786917767457283531353903875153871174800986390846405399072403710997
38858521698648340789796619739569871216543021087020838210840240445054
62887801800605087153155502339191346165512430620116086197691756282639
10889440556397873269527505542151344588992194802508034868063486617851
732353017537218467382336381147350936
p =
11208341989741288554420435511717274157087060775717507464810294560118
73317093331325798934485160437053238976589006339388863312279917506888
44975235155232055015971246527375860442003776715325329898599330337294
09611961422994399348900620060970672279330503181423390718558628104416
203494290159128272465260969051636856 1
Security Level = 512

This curve has a number of points ranging in the following interval:
Lower Bound =
11208341989741288554420435511717274157087060775717507464810294560118
73317093331325798934485160437053238976589006339388863312279917506888
44975235155232054994797355066672385347862471825954421398939042925921
57332419978943510079192764097528755549683840057783740811656017932251
208796848437132217231714573218607 4734
Upper Bound =
11208341989741288554420435511717274157087060775717507464810294560118
73317093331325798934485160437053238976589006339388863312279917506888
44975235155232055037145137988079335536145081604696238398259617748666
61891502867045288618608476024412589008977166305063040625461238276581
19819173188112432769880736488466623 90
Which are roughly 10^309 points

ECMQV w/ ECIES

A calcola le sue chiavi:
Privata statica:
28106839993234093186837310489027306687484724466412852
Pubblica statica: (-571506358, -1956162679)

Privata effimera: 413743053942312157093268505554
Pubblica effimera: (67601537, 1692777501)

B calcola le sue chiavi:
Privata statica: 150475
Pubblica statica: (-342292997, -1653381558)

Privata effimera: 64453138072285756552531982664390 20
Pubblica effimera: (-638081270, -713125219)

Chiave simmetrica calcolata da A: (1839074923, 806332297)
Chiave simmetrica calcolata da B: (1839074923, 806332297)

Cifratura e decifratura
Messaggio cifrato =
1f7dfdebe77dfde99f7dfde992c830677dfdeb3f7dfdef7dfde8aae77dfdef7dfde8
caa77dfde8203f7dfdef7dfdeba02a9b20d48a207a044babaa9bcab090b0a000d522
58f21993e40bd1ba504a803e4af0baeb4adb6114a564523212b7763
Messaggio decifrato = Prova del protocollo ECMQV usando cifratura
ECIES

ECMQV w/ ECAES

A calcola le sue chiavi:
Privata statica: 435874240368002579635389558022717496078088356
Pubblica statica: (-1739736326, 1311750026)

Privata effimera: 44378990169732168271187336736700 76
Pubblica effimera: (-1351205996, -2034421954)

B calcola le sue chiavi:
Privata statica:
78586594106360642615487436848653874649585525981188
Pubblica statica: (1836447438, 1981384197)

Privata effimera:
37047326109501461718515401779809694617845597675704313801
Pubblica effimera: (1029498896, -475309393)

Chiave simmetrica calcolata da A: (316861762, 364068364)
Chiave simmetrica calcolata da B: (316861762, 364068364)

Messaggio crittografato =
13cf7dfdeaff7dfdebb8677dfdef7dfde910ff7dfde83f7dfdef7dfdeb977dfde8c1
377dfdef7dfdebb1df7dfdeb1145f4404195d005c43141358400a525f0e090c4c8d2

3859324e4bdbfa9a8041f440f0fa21e01bc171405ea3529872f3dc8dccd84dcc0cce
59584e184e188e4dcd8cd88c9
Messaggio decifrato = Prova del protocollo ECMQV usando cifratura
ECAES