Hochschule Bonn-Rhein-Sieg

Angewandte Kryptographie 2, SoSe 2025

Vorlesungsteil "Kryptographie mit elliptischen Kurven"
Guntram Wicke (guntram.wicke@lehrbeauftragte.h-brs.de)

6 Praktikumsaufgaben "Elliptic Curve Cryptography (ECC)"

Hinweise: Abgabe der Lösungen bis zum 4.6.2025 auf LEA unter Angabe der Matrikelnummer(n) und Vorstellung am 6.6.2025.
Wenn Sie in Gruppen bis zu 3 Personen arbeiten: Eine Unteraufgabe hat genau eine Bearbeiterin / einen Bearbeiter. Z.B. eine Person implementiert in python, die andere testet in sage. Wechseln Sie ihre Rollen gleichmäßig.

6.1 Operationen DBL und ADD für Standard-projektive Koordinaten

1. (*python*) Implementieren Sie die Operation der Abelschen Gruppe für Punkte einer elliptischen Kurve, die in Standard-projektiven (homogenen) Koordinaten $(X : Y : Z)$ vorliegen. Die Domainparameter sind für eine Kurve in kurzer Weierstraß-Form über dem Grundkörper $GF(p)$ gegeben. Implementieren Sie zwei Funktionen in python, eine für die Punktaddition und eine für Punktverdopplung, und berücksichtigen Sie die Sonderfälle (welche sind das?).

   Die Koordinaten sind in python vom Typ `<type 'long'>`. In sage sind sie vom Typ `<class 'sage.rings.finite_rings.integer_mod.IntegerMod_gmp'>`

2. (*sage*) Testen Sie Ihre Implementierung exemplarisch mittels der Kurve brainpoolP256r1 mit folgenden Parametern:

```
ec_p = 0xA9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377
ec_a = 0x7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9
ec_b = 0x26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B6
ec_n = 0xA9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7
ec_gx= 0x8BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE27E1E3BD23C23A4453BD9ACE3262
ec_gy= 0x547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C1D54C72F046997
ec_h = 1
```

   Die Kurve und projektive Punkte darauf definieren Sie in *sage* wie folgt:

```
F = GF(ec_p)
E = EllipticCurve(F, [ec_a,ec_b])
P1= E([6030638041590466316856891123927382605 31\
448412342285592995176844173613464330 53,
746538570051509834695985451407074323090237 0296\
088143531902682622833903117 9596,
1])
P2 = 2*P1
P3 = P1 + P2
```

Ein Punkt wird durch `E()` als Element der definierten Kurve deklariert. Sollten die drei Koordinaten keinen Punkt definieren, wird ein Fehler generiert. (Werden nur zwei Koordinaten aufgelistet, wird der Punkt als in affinen Koordinaten gegeben akzeptiert.) Der neutrale Punkt wird durch `E(0)` erzeugt, was $(0 : 1 : 0)$ ergibt. Überlegen Sie sich eigene Tests für die Sonderfälle, insbesondere zwei gleiche und zwei zueinander inverse Punkte.

6.2 Punktmultiplikation ("Skalarmultiplikation")

1. (*python*) Implementieren Sie die Transformation von affiner Punktdarstellung (2 Koordinaten) in homogene projektive Darstellung (3 Koordinaten), sowie die inverse Transformation hierzu.

2. (*python*) Implementieren Sie die Punktmultiplikation $Q = [k]P$ für beliebige Punkte der Kurve, gegeben in affinen Koordinaten, und beliebige Skalare $k \geq 0$. Wählen sie einen Algorithmus ihrer Wahl aus der Vorlesung. Die Darstellung des Skalars sei als Langzahl, d.h. in Binärform msb first, nicht in NAF, gegeben. Berücksichtigen Sie, dass der Inputpunkt zuvor in projektive Koordinaten transformiert werden muss, und das Ergebnis ebenso rücktransformiert werden muss. Überlegen Sie sich ein sinnvolles Verhalten für den neutralen Punkt.

3. (*sage*) Testen Sie Ihre Implementierung, und nehmen Sie ggf. sage zu Hilfe. Berücksichtigen Sie insbesondere Grenzfälle und ungültige Punkte. Überlegen Sie, wann ein Punkt gültig oder ungültig ist.
   Zufällige Skalare x können Sie wie folgt in sage erzeugen, es muss noch eine Typkonvertierung mittels Integer() erfolgen:

```
N = GF(ec_n)
x = N.random_element()
Q = Integer(x)*P
```

6.3 ECDH

1. (*python*) Implementieren Sie das Schlüsseleinigungsprotokoll "Elliptic Curve Diffie Hellman". Es geht nicht um Netzwerkaspekte, sondern um Kryptographie. D.h., die Kommunikation zwischen Alice und Bob darf simuliert werden. Trennen sie dennoch zwischen den zwei Rollen, insbesondere was die Generierung der geheimen Schlüssel angeht (sie dürfen den Pseudozufall des Systems nutzen, es geht nur um einen Demonstrator.) Auf die authentische Übertragung der Nachrichten darf verzichtet werden.

   Nutzen Sie die Implementierung der Punktmultiplikation aus 6.2. Punkte werden in affinen Koordinaten (x, y) ausgetauscht, sie brauchen keine Punktkompression zu implementieren.

2. (*sage*) Vergleichen Sie die erzeugten Punkte der Pythonimplementierung mit Ergebnissen aus *sage*, die Sie z.B. wie folgt gewinnen:

```
PA = Integer(xa)*G
PB = Integer(xb)*G
ZA = Integer(xa)*PB
ZB = Integer(xb)*PA
ZA==ZB
True
```

3. (*python*) Überlegen sie sich einen Man/Woman-in-the-middle-Angriff und implementieren Sie ihn. Überzeugen Sie sich, dass Charlie/Eve jeweils gemeinsame Geheimnisse mit Alice und Bob etablieren können, wenn kein authentischer Nachrichtenaustausch erfolgt.

4. (*python*) Etablieren Sie mittels Ihrer getesteten Pythonimplementierung gemeinsame Geheimnisse mit den anderen Gruppen des Praktikums. Sie können der einfachen Prüfbarkeit wegen die Kurve $y^2 = x^3 + 16x + 20 \, mod \, 31$ nehmen.