Supersingular Isogeny Elliptic Curve Cryptography

Before we start, let's be clear: this is an experiment to demo isogeny-based DH, it is not secure or fast (at least it wouldn't be with reasonably-sized fields)!

We pick a supersingular curve over a small prime field:

```
lA, lB = 2, 3
eA, eB = 6, 7
p = lA ^ eA * lB ^ eB - 1 # This is conveniently a large-ish curve
for a demo (comically small for crypto, though!); this structure
doesn't matter much because we do math over GF(p), not GF(p^2)
assert p.is_prime()
assert p % 4 == 3 # Necessary for below curve to be supersingular.
```

```
GF(p^2)
```

```
    Finite Field in z2 of size 139967^2
```

```
k = GF(p) # Note; not using GF(p^2) because of a limitation in Sage
E = EllipticCurve(k, [1, 0])
E
```

```
    Elliptic Curve defined by y^2 = x^3 + x over Finite Field of size
    139967
```

Elliptic curves of this form with a prime congruent to 3 mod 4 will incidentally always be supersingular, but Sage will confirm that:

```
E.is_supersingular()
```

```
    True
```

```
n_points = E.count_points()
n_points
```

```
    139968
```

```
E.j_invariant()
```

```
    1728
```

Let's pick 4 random unique points, fixed as part of the protocol:

```
points = []
while len(points) != 4:
    p = E.random_point()
    if p not in points:
        points.append(p)
```

```
PA, PB, QA, QB = points
PA, PB, QA, QB
```

```
((129731 : 133310 : 1),
 (89516 : 39263 : 1),
 (75830 : 10281 : 1),
 (4425 : 63959 : 1))
```

Alice computes her secret numbers, from which she computes a point RA, which defines the kernel of her isogeny:

```
mA, nA = 123, 525
RA = mA * PA + nA * QA
print RA
phiA = E.isogeny(RA)
EA = phiA.codomain()
```

```
(134960 : 51025 : 1)
```

Sage has convenient tools for proving that this is an isogeny:

```
E.is_isogenous(EA)
```

```
True
```

Alice sends her public key (consisting of the isogenous elliptic curve and the two base points for Bob under that curve) to Bob. I use the symbols phiA_PB and phiA_QB here to clarify that Bob just sees those values; he does not actually see the isogeny itself.

```
EA, phiA_PB, phiA_QB = EA, phiA(PB), phiA(QB)
EA, phiA_PB, phiA_QB
```

```
(Elliptic Curve defined by y^2 = x^3 + 130855*x + 32368 over Finit
Field of size 139967,
 (651 : 40521 : 1),
 (1728 : 0 : 1))
```

Bob does the same thing:

```
# Bob does the same thing
mB, nB = 812, 580
RB = mB * PB + nB * QB
print RB

# phiB is a function from points on E to points on EB
phiB = E.isogeny(RB)
print phiB
EB = phiB.codomain()
print EB
```

```
(36575 : 8140 : 1)
Isogeny of degree 34992 from Elliptic Curve defined by y^2 = x^3 +
over Finite Field of size 139967 to Elliptic Curve defined by y^2
```

```
x^3 + 115910*x + 38819 over Finite Field of size 139967
Elliptic Curve defined by y^2 = x^3 + 115910*x + 38819 over Finite
Field of size 139967
```

```
E.is_isogenous(EB)
```

```
True
```

```
# Bob sends to Alice:
EB, phiB_PA, phiB_QA = EB, phiB(PA), phiB(QA)
EB, phiB_PA, phiB_QA
```

```
(Elliptic Curve defined by y^2 = x^3 + 115910*x + 38819 over Finit
Field of size 139967,
 (17496 : 82589 : 1),
 (17496 : 57378 : 1))
```

```
# Alice computes the shared secret:
SBA = mA * phiB_PA + nA * phiB_QA
print SBA
phiBA = EB.isogeny(SBA)
print phiBA
KA = phiBA.codomain().j_invariant()
```

```
(34992 : 0 : 1)
Isogeny of degree 2 from Elliptic Curve defined by y^2 = x^3 +
115910*x + 38819 over Finite Field of size 139967 to Elliptic Curv
defined by y^2 = x^3 + 104975*x over Finite Field of size 139967
```

```
# Bob computes the shared secret:
SAB = mB * phiA_PB + nB * phiA_QB
print SAB
phiAB = EA.isogeny(SAB)
print phiB
KB = phiAB.codomain().j_invariant()
```

```
(651 : 99446 : 1)
Isogeny of degree 34992 from Elliptic Curve defined by y^2 = x^3 +
over Finite Field of size 139967 to Elliptic Curve defined by y^2
x^3 + 115910*x + 38819 over Finite Field of size 139967
```

```
KA == KB
```

```
True
```