CS 259C/Math 250: Elliptic Curves in Cryptography
Homework 2 Solutions

1. (a) We are given that the equation for $E^{(d)}$ is

$$y^2 = x^3 + Ad^2x + Bd^3$$

Dividing by d^3 gives

$$\frac{y^2}{d^3} = \frac{x^3}{d^3} + \frac{Ax}{d} + B$$

Which can be rewritten as

$$d\left(\frac{y}{d^2}\right)^2 = \left(\frac{x}{d}\right)^3 + A\left(\frac{x}{d}\right) + B$$

The the transformation $\left(\frac{x}{d}, \frac{y}{d^2}\right) \to (x, y)$ gives us the desired transformation.

(b) The equation for E is

$$y_1^2 = x_1^3 + Ax_1 + B$$

Let g be a generator for $\mathbb{F}_q^\times$. Then $D = g^{n'}$ for some n'. Since D is not a square, it must be that n' is odd, so $D = g^{2n+1}$ for some n. The equation for $E' = E^{(D)}$ is

$$Dy_2^2 = x_2^3 + Ax_2 + B$$

Similarly, $d = g^{m'}$ for some m', and the equation for $E^{(d)}$ is

$$dy_3^2 = x_3^3 + Ax_3 + B$$

There are two cases:

* d is a square. In this case, m' is even, so we can write $d = g^{2m}$. Let $x_1 = x_3$ and $y_1 = g^m y_3$. Then:

$$y_1^2 = g^{2m}y_3^2 = dy_3^2 = x_3^3 + Ax_3 + B = x_1^3 + Ax_1 + B$$

Thus, the curve is isomorphic to E.

* d is not a square. In this case, m' is odd, so we can write $d = g^{2m+1}$. Let $x_2 = x_3$ and $y_2 = g^{m-n}y^3$. Then:

$$Dy_2^2 = g^{2n+1}g^{2m-2n}y_3^2 = g^{2m+1}y_3^2 = dy_3^2 = x_3^3 + Ax_3 + B = x_2^3 + Ax_2 + B$$

Thus, the curve is isomorphic to E'.

(c) Let $\#E(\mathbb{F}_q) = q + 1 - t$ and and $\#E'(\mathbb{F}_q) = q + 1 - t'$. According to a slight generalization of Theorem 4.14, the number of points in $\mathbb{F}_q \times \mathbb{F}_q$ on the curve defined by $y^2 = f(x)$ is

$$q + 1 + \sum_{x \in \mathbb{F}_q} \left(\frac{f(x)}{\mathbb{F}_q} \right)$$

Where $\left(\frac{z}{\mathbb{F}_q}\right)$ is the Legendre symbol (0 if $z = 0$, 1 if z is a square, and -1 otherwise). Since E is specified by

$$y^2 = x^3 + Ax + B,$$

this means

$$t = -\sum_{x \in \mathbb{F}_q} \left(\frac{x^3 + Ax + B}{\mathbb{F}_q} \right)$$

Since E' is specified by

$$y^2 = \frac{x^3 + Ax + B}{D}$$

for a non-square D,

$$t' = -\sum_{x \in \mathbb{F}_q} \left(\frac{(x^3 + Ax + B)/D}{\mathbb{F}_q} \right)$$

Note that if $z = 0$, so is z/D; if z is a square, z/D is not; and if z is not a square, z/D is. Therefore,

$$\left(\frac{x^3 + Ax + B}{\mathbb{F}_q} \right) = -\left(\frac{(x^3 + Ax + B)/D}{\mathbb{F}_q} \right)$$

Summing over all $x \in \mathbb{F}_q$ gives us $t = -t'$, as desired.

(d) Recall that

$$\#E(\mathbb{F}_{q^n}) = (q^n + 1) - (\alpha^n + \beta^n)$$

Where α and β are solutions to $X^2 - tX + q = 0$. Note that $\alpha + \beta = t$ and $\alpha\beta = q$. Thus:

$$\begin{aligned} \#E(\mathbb{F}_{q^2}) &= (q^2 + 1) - (\alpha^2 + \beta^2) = q^2 + 1 - (\alpha + \beta)^2 + 2\alpha\beta \\ &= q^2 + 1 - t^2 + 2q = (q + 1 - t)(q + 1 + t) = \#E(\mathbb{F}_q)\#E'(\mathbb{F}_q) \end{aligned}$$

(e)

```
E=EllipticCurve(GF(5),[1,0]); E.abelian_group()
```

```
Additive abelian group isomorphic to Z/2 + Z/2 embedded in Abelian group
of points on Elliptic Curve defined by y^2 = x^3 + x over Finite Field
of size 5
```

```
E_twist=E.quadratic_twist(); E_twist.abelian_group()
```

```
Additive abelian group isomorphic to Z/2 + Z/4 embedded in Abelian group
of points on Elliptic Curve defined by y^2 = x^3 + 4*x over Finite Field
of size 5
```

```
E2=EllipticCurve(GF(25,'a'),[1,0]); E2.abelian_group()
```

```
Additive abelian group isomorphic to Z/4 + Z/8 embedded in Abelian group
of points on Elliptic Curve defined by y^2 = x^3 + x over Finite Field
in a of size 5^2
```

$$E(\mathbb{F}_q) \times E'(\mathbb{F}_q) = (\mathbb{Z}_2 \times \mathbb{Z}_2) \times (\mathbb{Z}_2 \times \mathbb{Z}_4) \ncong \mathbb{Z}_4 \times \mathbb{Z}_8 = E(\mathbb{F}_{q^2})$$

Thus we have found a counterexample.

2. Since -1 is a non-square in $\mathbb{F}_p$ when $p \equiv 3 (\text{mod } 4)$, the quadratic twist of E, E' can be given by

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

This is equivalent to

$$y^2 = -(x^3 + Ax) = (-x)^3 + A(-x)$$

Thus we have an isomorphism between E and E' given by $x \to -x$. Thus, $\#E(\mathbb{F}_p) = \#E'(\mathbb{F}_p)$. By question 1 part (c), we have

$$p + 1 - t = \#E(\mathbb{F}_p) = \#E'(\mathbb{F}_p) = p + 1 + t$$

So $t = 0$. This means $\#E(\mathbb{F}_p) = p + 1$, and therefore E is supersingular.

3. (a) Recall that ϕ satisfies the equation $X^2 - tX + p = 0$, where $\#E(\mathbb{F}_p) = p + 1 - t$. Since E is supersingular, $\#E(\mathbb{F}_p) = p+1$, so $t = 0$. Hence $\phi^2+p = \phi^2+t\phi+p = 0$.

   (b) If $(x, y) \in E[p + 1]$, then $[p + 1](x, y) = 0$, or subtracting $[p](x, y)$ gives $(x, y) = [-p](x, y)$. But according to part (a), $\phi^2(x, y) = [-p](x, y)$. Thus, if $(x, y) \in E[p + 1]$, $\phi^2(x, y) = [-p](x, y) = (x, y)$, so ϕ^2 is the identity on $E[p + 1]$.

   (c) As mentioned above, $t = 0$. Thus $\#E'(\mathbb{F}_p) = q + 1 + t = q + 1$. This means

$$\#E(\mathbb{F}_{p^2}) = \#E(\mathbb{F}_p)\#E'(\mathbb{F}_p) = (p + 1)(p + 1) = p^2 + 2p + 1$$

   (d) According to Lemma 4.5, since ϕ^2 (which is the Frobenius endomorphism for $\mathbb{F}_{p^2}$) is the identity on all points in $E[p+1]$, $E[p+1] \subset E(\mathbb{F}_{p^2})$. According to Theorem 3.1,

$$E[p + 1] = \mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$$

Meaning that $E[p + 1]$ has $(p + 1)^2 = p^2 + 2p + 1$ elements, the same as $E(\mathbb{F}_{p^2})$. Thus

$$E(\mathbb{F}_{p^2}) = E[p + 1] = \mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$$

(e) The fact given reads: For any $n \geq 2$, $p \nmid n$, if $E[n] \subset E(\mathbb{F}_p)$, then $n|(p-1)$. Theorem 4.1 tells us that

$$E(\mathbb{F}_p) = \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$$

Where n_1 divides n_2. If $n_1 = 1$ we are done. Otherwise, since $\#E(\mathbb{F}_p) = p+1$, we have $n_1 n_2 = p+1$, and thus $n_1|(p+1)$. Also, $E[n_1] = \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_1}$ is a subset of $E(\mathbb{F}_p)$. But now, according to the fact, n_1 divides $p-1$. The only way for $n_1 \neq 1$ to divide both $p-1$ and $p+1$ is for n_1 to be 2. This proves the $p \equiv 3 (\text{mod } 4)$ case.

If $p \equiv 1 (\text{mod } 4)$, then $p+1 \equiv 2 (\text{mod } 4)$. This means that $p+1$ is even, but it is not divisible by 4. However, if $n_1 = 2$, then $2|n_2$, so $p+1 = n_1 n_2$ is divisible by 4. Therefore, $n_1 = 1$ is the only possibility, so $E(\mathbb{F}_p)$ is cyclic.

4. (a) $(1+\alpha)(x,y) = (x,y) + \alpha(x,y) = (x,y) + (-x, iy)$. Using the group law for elliptic curves, $(1+\alpha)(x,y) = (x_1, y_1)$ where:

$$\begin{aligned} x_1 &= m^2 - x + x = m^2 \\ y_1 &= m(x - x_3) + y \\ m &= \frac{iy - y}{-x - x} = \frac{1-i}{2}\frac{y}{x} \end{aligned}$$

Thus,

$$\begin{aligned} (1+\alpha)(x,y) &= \left(\left(\frac{1-i}{2}\right)^2 \frac{y^2}{x^2}, \frac{1-i}{2}\frac{y}{x}\left(x - \left(\frac{1-i}{2}\right)^2 \frac{y^2}{x^2}\right) - y \right) \\ &= \left(-\frac{i}{2}\frac{y^2}{x^2}, \left(\frac{1-i}{2} - \left(\frac{1-i}{2}\right)^3 \frac{y^2}{x^3} + 1\right) y \right) \\ &= \left(-\frac{i}{2}\frac{x^3 + Ax}{x^2}, \left(\frac{3-i}{2} + \frac{1+i}{4}\frac{x^3 + Ax}{x^3}\right) y \right) \\ &= \left(-\frac{i}{2}\frac{x^2 + A}{x}, \frac{1+I}{4}\frac{(3-4i)x^2 + A}{x^2} y \right) \end{aligned}$$

Therefore, $1+\alpha$ has degree 2.

(b) We can compute the kernel in two ways. The first is to notice that the only (finite) points that map to infinity according to the calculation in part (a) are those with $x = 0$. This implies $y = 0$. Thus, the kernel is $\{\infty, (0,0)\}$. Alternatively, we can observe that (x,y) is in the kernel of $1+\alpha$ if $\alpha(x,y) = -(x,y)$. This is equivalent to

$$(-x, iy) = (x, -y)$$

This implies $2x = 0$. Since we are not in characteristic 2, we must have $x = 0$, which implies $y = 0$. Thus the kernel is $\{\infty, (0,0)\}$

(c) Proposition 3.16 reads:

$$\deg(a'\alpha' + b'\beta') = a'^2 \deg \alpha' + b'^2 \deg \beta' + a'b'(\deg(\alpha' + \beta') - \deg \alpha' - \deg \beta')$$

Setting $a' = a$, $\alpha' = 1$, $b' = b$, $\beta' = \alpha$ gives us:

$$\deg([a]+[b]\alpha) = \deg(a[1]+b\alpha) = a^2 \deg 1 + b^2 \deg \alpha + ab(\deg(1+\alpha) - \deg 1 - \deg \alpha$$

We compute $\deg(1+\alpha)$ in part (a) to be 2, and $\deg \alpha = 1$ and $\deg 1 = 1$ since they are linear. Thus:

$$\deg([a]+[b]\alpha) = a^2 + b^2 + ab(2-1-1) = a^2 + b^2$$

(d) Note that α is an endomorphisms, so it commutes with $[n]$ for all n. Also note that $\alpha^2(x,y) = \alpha(-x, iy) = (x, -y) = -(x,y)$. Thus,

$$([a]-[b]\alpha)([a]+[b]\alpha) = [a]^2 + [a][b]\alpha - [b]\alpha[a] - [b]\alpha[b]\alpha = [a^2] + [b^2] = [a^2+b^2]$$

This means that if $\phi_{a,b} = [a]+[b]\alpha$ and $\psi_{a,b} = [a]-[b]\alpha$, then $\psi_{a,b}\phi_{a,b} = [a^2+b^2]$. Similarly, $\phi_{a,b}\psi_{a,b} = [a^2+b^2]$

5. (a) Let P_1, Q_1 be generators for $E(\mathbb{F}_p) \cong \mathbb{Z}_{26} \times \mathbb{Z}_{26}$. We can infer that P_1 and Q_1 have order 26, since otherwise, $E(\mathbb{F}_p)$ would not be isomorphic to $\mathbb{Z}_{26} \times \mathbb{Z}_{26}$. The subgroup of elements of order 13 will then be generated by $P = 2P_1, Q = 2Q_1$. There will be $13^2 = 169$ elements generated by P and Q. According to Theorem 3.2, since $677 \nmid 13$, $E[13] \cong \mathbb{Z}_{13} \times \mathbb{Z}_{13}$, which has $13^2 = 169$ elements. This means all the elements of order 13 are in fact in $E(\mathbb{F}_p)$, and thus a pair for generators for $E[13]$ is given by P, Q. Here is the sage code for this:

```
E=EllipticCurve(GF(677),[1,0])
gens=[i+i for i in E.gens()]
P=gens[0]
Q=gens[1]
print(gens)
```

```
[(388 : 417 : 1), (350 : 246 : 1)]
```

(b) Note that α was defined as $\alpha(x,y) = (-x, iy)$, where i is a square root of -1. We note that $-1 \equiv 676$ in $\mathbb{F}_{677}$, and $26^2 = 676$. Thus, the square roots of -1 lie in $\mathbb{F}_{677}$, and hence α is an automorphism on $E(\mathbb{F}_p)$. We can use Sage to list the automorphisms:

```
auts=E.automorphisms();auts
```

```
[Generic endomorphism of Abelian group of points on Elliptic Curve
defined by y^2 = x^3 + x over Finite Field of size 677
  Via:  (u,r,s,t) = (1, 0, 0, 0), Generic endomorphism of Abelian group
of points on Elliptic Curve defined by y^2 = x^3 + x over Finite Field
of size 677
  Via:  (u,r,s,t) = (26, 0, 0, 0), Generic endomorphism of Abelian group
of points on Elliptic Curve defined by y^2 = x^3 + x over Finite Field
of size 677
  Via:  (u,r,s,t) = (651, 0, 0, 0), Generic endomorphism of Abelian
group of points on Elliptic Curve defined by y^2 = x^3 + x over Finite
Field of size 677
  Via:  (u,r,s,t) = (676, 0, 0, 0)]
```

Notice that the first automorphisms is the identity, and the last is the inverse automorphisms. Thus, the other two must correspond to our αs (one for $i = -26 =$

651, the other for $i = 26$). We can choose either one to be our automorphism. I will chose the first.

To compute the action of α, we simply compute $\alpha(P)$ and $\alpha(Q)$, and iterate over all linear combinations of P and Q until we find the combinations equal to $\alpha(P)$ and $\alpha(Q)$. Here is the code:

```
aut=auts[1]
for a in range(P.order()):
    for b in range(Q.order()):
        t = a * P + b * Q
        if aut(P)==t:
            print 'aut(P) = '+str(a)+'P + '+str(b)+'Q'
            M11 = a
            M21 = b
        if aut(Q)==t:
            print 'aut(Q) = '+str(a)+'P + '+str(b)+'Q'
            M12 = a
            M22 = b
M=matrix(GF(13),[[M11,M12],[M21,M22]])
print(M)
```

```
aut(P) = 4P + 1Q
aut(Q) = 9P + 9Q
[4 9]
[1 9]
```

(c) Again, Sage provides the answer:

```
M*M
```

```
[12  0]
[ 0 12]
```

Since $12 = -1$ in $\mathbb{F}_{13}$, this is the desired answer.

(d) We can use Sage to compute the eigensystem:

```
eigensys = M.eigenvectors_right(); print(eigensys)
```

```
[(8, [
(1, 12)
], 1), (5, [
(1, 3)
], 1)]
```

This tells us that the eigenvalues are 5 and 8 (which square to -1 in $\mathbb{F}_{13}$), and also tells us what the corresponding eigenvectors are. We can then define PP and QQ using this information, and check that they are eigenvectors of α:

```
lambda1 = (eigensys[0][0]).lift()
lambda2 = (eigensys[1][0]).lift()
PP=(eigensys[0][1][0][0]).lift()*P+(eigensys[0][1][0][1]).lift()*Q
QQ=(eigensys[1][1][0][0]).lift()*P+(eigensys[1][1][0][1]).lift()*Q
print(lambda1)
print(lambda2)
print(PP)
print(QQ)
```

```
8
5
(558 : 287 : 1)
(95 : 442 : 1)
```

```
print(aut(PP)==lambda1*PP)
print(aut(QQ)==lambda2*QQ)
```

```
True
True
```

6. (a) Suppose there is an algorithm $\mathcal{A}$ that has advantage ϵ in the semantic security game. Define two algorithms, $\mathcal{A}_0$ and $\mathcal{A}_1$, which play the real-or-zero game. $\mathcal{A}_b$ works as follows: on input pk, simulate $\mathcal{A}$ on pk. When $\mathcal{A}$ produces a challenge query (m_0, m_1), submit $m = m_b$ are the challenge query to the real-or-zero challenger. Send the response c back to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}_b$ returns b'. Observe that

$$\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)] = \Pr[\mathcal{A}_b(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)]$$

Also, note that

$$\Pr[\mathcal{A}_0(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)] = \Pr[\mathcal{A}_1(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)]$$

Then we have:

$$\begin{aligned}
\epsilon &= |\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0)] - \Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1)]| \\
&= |\Pr[\mathcal{A}_0(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)] - \Pr[\mathcal{A}_1(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)]| \\
&= |\Pr[\mathcal{A}_0(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)] - \Pr[\mathcal{A}_0(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)] \\
&\quad + \Pr[\mathcal{A}_1(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)] - \Pr[\mathcal{A}_1(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)]| \\
&\leq || \Pr[\mathcal{A}_0(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)] - \Pr[\mathcal{A}_0(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)]| \\
&\quad + |\Pr[\mathcal{A}_1(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)] - \Pr[\mathcal{A}_1(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)]| \\
&= \epsilon_0 + \epsilon_1
\end{aligned}$$

Where ϵ_b is the advantage of $\mathcal{A}_b$. At least one b has $\epsilon_b \geq \frac{\epsilon}{2}$, so let $A' = A_b$ for that b. Then A' has advantage at least $\epsilon/2$.

(b) Suppose there is an adversary $\mathcal{A}'$ that has advantage ϵ in the real-or-zero game. Define $\mathcal{A}$ to be the following algorithm: on input pk, simulate $\mathcal{A}'$ on pk. When $\mathcal{A}'$ produces the challenge query m, set $m_0 = m$, and $m_1 = 0$, and send (m_0, m_1) to the semantic security challenger. Send the response c back to $\mathcal{A}'$. When $\mathcal{A}'$ produces a bit b', output b'. We have that:

$$\begin{aligned}
\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0)] &= \Pr[\mathcal{A}'(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)] \\
\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1)] &= \Pr[\mathcal{A}'(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)]
\end{aligned}$$

Thus,

$$\begin{aligned}\epsilon &= |\Pr[\mathcal{A}'(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)] - \Pr[\mathcal{A}'(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(0)]| \\ &= |\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0)] - \Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1)]|\end{aligned}$$

Which is exactly the advantage of $\mathcal{A}$. Therefore, $\mathcal{A}$ has advantage ϵ.

(c) $\mathcal{E}'$ is just $\mathcal{E}$ with a zero appended to the end. Let $\mathcal{A}$ be an adversary with advantage at least ϵ against $\mathcal{E}'$. Let $\mathcal{B}$ be the following algorithm that breaks $\mathcal{E}$: on input pk, simulate $\mathcal{A}$ on pk. When $\mathcal{A}$ produces the challenge query, $\mathcal{B}$ forwards that query to its challenger. When the challenger responds with a ciphertext c, $\mathcal{B}$ responds to $\mathcal{A}$ with $c||0$. When $\mathcal{A}$ outputs b', $\mathcal{B}$ outputs b'. It is clear that when $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$, then $c||0$ is distributed according to $\mathsf{Enc}'_{\mathsf{pk}}(m)$ Hence, $\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}'_{\mathsf{pk}}(m_b) = \Pr[\mathcal{B}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)]$ It easily follows that $\mathcal{B}$ has the same advantage as $\mathcal{A}$.

(d) The idea here is that a random element of the ciphertext space is not the same as an encryption of a random message from the message space, as in the real-or-random game. It is actually distributed quite differently as the last bit of a random element of the ciphertext space is 0 or 1 with equal probability, but the encryption of any message always ends in a 0. Thus, we get a simple adversary for $\mathcal{E}'$: on input pk, send 0 as the challenge query. When the challenger responds with $c||b'$, return b'. The advantage of $\mathcal{A}$ is:

$$\left|\Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathsf{Enc}'_{\mathsf{pk}}(0)] - \Pr[\mathcal{A}(\mathsf{pk}, c) = 1 : c \leftarrow \mathcal{C}]\right| = \left|0 - \frac{1}{2}\right| = \frac{1}{2}$$

7. First, we need to show that we can always detect if an element x is a square in $\mathbb{G}$. To do this, compute x^r, and check if it equals 1. If x is a square, then $x = g^{2a}$ for some a, so $x^r = (g^{2r})^a = 1$. If x is not a square, then $x = g^{2a+1}$ for some a, so $x^r = (g^{2r})^a g^r = g^r \neq 1$.

   Now, let $A(g, x_1, x_2, x_3)$ be the following algorithm: output 1 if and only if either

   - x_1 or x_2 are squares, and so is x_3.
   - x_1, x_2 and x_3 are all non-squares.

   Now let's analyse the two cases in the Decisional Diffie-Hellman problem:

   - (g, g^a, g^b, g^{ab}) for random $a, b \in [0, 2r-1]$. If either g^a or g^b are squares, then a or b are even, so ab is even, and thus g^{ab} is a square. If neither g_a or g_b are squares, then both a and b are odd, so ab is odd, and thus g^{ab} is not a square. Thus A will always output 1 in this case.
   - (g, g^a, g^b, g^c) for random $a, b, c \in [0, 2r-1]$. g^c will be a square or non-square with probability $\frac{1}{2}$, independent of g^a and g^b. Thus the probability that A outputs 1 is exactly $\frac{1}{2}$.

Hence, the advantage of A in the Decisional Diffie-Hellman problem is exactly $\frac{1}{2}$.

In general, if the order of $\mathbb{G}$ factors as fr' for some small prime factor f, we can always test if an element x has an fth root by checking if $x^{r'} = 1$. Then, we can write an algorithm $A(g, x_1, x_2, x_3)$ which checks if either x_1 or x_2 have fth roots and x_3 has an fth root, or if none of them do. When the input is (g, g^a, g^b, g^{ab}), the output will always be 1. When the input is (g, g^a, g^b, g^c), it can be shown that the probability A outputs 1 is less than or equal to $1 - \frac{1}{f}$ for all $f \geq 2$. Thus, A has advantage $\frac{1}{f}$.

8. (a) Recall that to generate a random point on the curve, we pick a random $x \in \mathbb{F}_p$ and compute $z = x^3 + Ax + B$. If z is a square, we output one of the roots. Otherwise, we start over. We can use this idea to encode an integer n as a point on the curve: divide the domain of x into buckets, one for each meassage in the message space. To encode a message m, pick an x from the bucket corresponding to m. If $x^3 + Ax + B$ has a square root y, return (x, y). Otherwise generate another x in the bucket and try again. If $m \in [0, N)$, one approach is to fix an r, and the bucket corresponding to m is $[mr, (m+1)r)$. We can generate elements in the bucket in two ways: either iterate through the elements deterministically, or randomly pick an element. The deterministic way will be a little faster (since we will not generate the same element twice). However, the random way will make a brute-force attack more difficult since it expands the number possible ciphertexts for each message.

   The inverse map takes a point (x, y) and returns $\lfloor x/r \rfloor$ Since $x \in [mr, (m+1)r)$, we have that $x/r \in [m, m+1)$, so taking the floor gives m, as desired.

   We need to pick r large enough so that each bucket has at least one x that gives rise to a point on the curve. Since there are N buckets of size r, we need $Nr \leq p$. Hence we can choose $r = \left\lfloor \frac{p}{N} \right\rfloor$. We get the following algorithm:

```
def int_to_point(n,E):
    '''Takes an integer n and encodes it as a unique point on the elliptic curve E.
    '''
    r=floor(E.base_field().order()/36^25)
    while True:
        x=r * n + ZZ.random_element(r)
        zs=E.lift_x(x,True)
        l=len(zs)
        if l==0:
            continue
        i=ZZ.random_element(l)
        P=zs[i]
        return P
```

   Note that `E.lift_x` takes a value x, and tries to generate points (x, y) on E. The second argument specifies whether or not to generate the entire list of points, or just one point (if it exists).

   (b)

```
def point_to_int(P):
    '''Takes a point P on an elliptic curve and encodes it as an integer.
    '''
    r=floor(P.curve().base_field().order()/36^25)
    x=P.xy()[0]
    return floor(x.lift()/r)
```

Note that `x.lift()` takes x (which is an element of $\mathbb{F}_q$), and returns the equivalent integer in $\mathbb{Z}$.

(c)

```
def encrypt(P,Q,s):
    '''Compute the ElGamal encryption of the string s using the public key (P,Q).'''

    y = ZZ.random_element(P.order())
    return (y*P,int_to_point(str_to_int(s),P.curve())+y*Q)
```

(d)

```
encrypt(P,Q,'mark')
```

```
((3705220128636504713390744639982221516765029051814359661997 :
335635218641073950616862297863741970869847882781745316502 8 : 1),
(291709547023462266756809712886301699955207224258050128 17 :
500654738637751108806922172119850551217710871570932488579 8 : 1))
```

(e) This is the NIST P-192 curve. It was chosen because the group order is large and prime, and P has the same order as the group. Also, the field size p is chosen so that arithmetic in $\mathbb{F}_p$ is very fast. For example, reductions mod p can be computed at the level of 64-bit words. For more information, go to this site and download `fips_186-3.pdf`. The curve is defined on page 88.

9. The Schnorr signatures will be:

   - $(R, s) = ([k_i]P, k_i + ae (\text{mod } r))$
   - $(R', s') = ([k_{i+1}]P, k_{i+1} + ae' (\text{mod } r)$

Where $e = H(M||R)$ and $e' = H(M'||R')$, M, and M' are the two signed messages, and k_i, k_{i+1} is our bad randomness. Since $k_{i+1} = Ak_i + B$, we have

$$s' = Ak_i + B + ae' (\text{mod } r)$$

Thus, assuming $Ae \neq e' (\text{mod } r)$, we have

$$\frac{As + B - s'}{Ae - e'} = \frac{A(k_i + ae) + B - (Ak_i + B + ae')}{Ae - e'} = \frac{a(Ae - e')}{Ae - e'} = a (\text{mod } r)$$

If $Ae = e' (\text{mod } r)$, then $AH(M||R) = H(M'||R')$ mod r. If $M \neq M'$, then the inputs are different, so the outputs satisfy $Ae = e'$ with probability $1/r$ assuming a truly random oracle H. If $M = M'$, then the inputs are different ($R \neq R'$) with probability $1 - 1/r$. In this case, the outputs satisfy $Ae = e'$ with probability $1/r$. Thus the probability that we succeed is at least $(1-1/r)^2$. Either way, the probability is at least $(1 - 1/r)^2$, which is very close to 1.