CS 259C/Math 250: Elliptic Curves in Cryptography
Homework 3 Solutions

1. (a)

$$\begin{aligned}
\mathsf{Enc}(\mathsf{pp}, Q_A, Q_B, M) &= ([c]P, M\gamma) \text{ where } c \xleftarrow{R} [1, n] \text{ and } \gamma = \hat{e}(Q_A, Q_B)^c \\
\mathsf{Dec}_A(\mathsf{pp}, a, (c_1, c_2)) &= c_2\gamma_A^{-1} \text{ where } \gamma_A = \hat{e}(c_1, Q_B)^a \\
\mathsf{Dec}_B(\mathsf{pp}, b, (c_1, c_2)) &= c_2\gamma_B^{-1} \text{ where } \gamma_B = \hat{e}(Q_A, c_1)^b
\end{aligned}$$

If $(c_1, c_2) = ([c]P, M\gamma)$ where $\gamma = \hat{e}(Q_A, Q_B)^c$, then

$$\begin{aligned}
\gamma &= \hat{e}(Q_A, Q_B)^c = \hat{e}([a]P, [b]P)^c = \hat{e}(P, P)^{abc} \\
\gamma_A &= \hat{e}(c_1, Q_B)^a = \hat{e}([c]P, [b]P)^a = \hat{e}(P, P)^{abc} = \gamma \\
\gamma_B &= \hat{e}(Q_A, c_1)^b = \hat{e}([a]P, [c]P)^b = \hat{e}(P, P)^{abc} = \gamma
\end{aligned}$$

Thus, both decryption algorithms correctly recover M

(b) Suppose there is an adversary $\mathcal{A}$ with advantage at least ϵ in the real-or-random game. Then we can construct the following algorithm B which solves the BDDH problem:

* On input (P, Q, R, S, γ), send (P, Q, R) to $\mathcal{A}$, where Q is interpreted as Q_A and R is interpreted as Q_B.
* When A send the challenge plaintext M, respond with $(S, M\gamma)$ as the challenge ciphertext.
* Output the output of $\mathcal{A}$

Since $Q = Q_A = [a]P$ and $R = Q_B = [b]P$ for random a, b, the public parameters seen by $\mathcal{A}$ are distributed as they would be in the real-or-random game. If $\gamma = \hat{e}(P, P)^{abc}$ where $S = [c]P$, then the challenge ciphertext $(S, M\gamma) = ([c]P, M\gamma)$ is a correctly distributed encryption of M since c is random. Hence,

$$\Pr[\mathcal{B} = 1 | \gamma = \hat{e}(P, P)^{abc}] = \Pr[\mathcal{A} = 1 | \text{real encryption}]$$

If γ is random, then denote $M' = M\gamma\hat{e}(P, P)^{-abc}$. Since γ is random, M' is a random message. Further, $(S, M\gamma) = ([c]P, M'\hat{e}(P, P)^{abc})$, which is a correctly distributed encryption of the random message M'. Hence,

$$\Pr[\mathcal{B} = 1 | \gamma \xleftarrow{R} \mu_n] = \Pr[\mathcal{A} = 1 | \text{random encryption}]$$

Thus, the advantage of $\mathcal{B}$ in the BDDH game is identical to the advantage of $\mathcal{A}$ in the real-or-random game. Specifically, the advantage is at least ϵ.

(c)

$$\begin{aligned}\mathsf{Enc}(\mathsf{pp}, Q_1, ..., Q_\ell, M) &= ([r]P, M, \gamma) \text{ where } r \xleftarrow{R} [1, n] \text{ and } \gamma = \hat{e}(Q_1, ..., Q_\ell)^r \\ \mathsf{Dec}_i(\mathsf{pp}, a_i, (c_1, c_2)) &= c_2\gamma_i^{-1} \text{ where } \gamma_i = \hat{e}(Q_1, ..., Q_{i-1}, c_1, Q_{i+1}, ..., Q_\ell)^{a_i}\end{aligned}$$

If $(c_1, c_2) = ([r]P, M\gamma)$ where $\gamma = \hat{e}(Q_1, ..., Q_\ell)^r$, then

$$\begin{aligned}\gamma &= \hat{e}(Q_1, ..., Q_\ell)^r = \hat{e}(P, ..., P)^{a_1 ... a_\ell r} \\ \gamma_i &= \hat{e}(Q_1, ..., Q_{i-1}, [r]P, Q_{i+1}, ..., Q_\ell)^{a_i} = \hat{e}(P, ..., P)^{a_1 ... a_\ell r} = \gamma\end{aligned}$$

Thus all decryption algorithms correctly recover M.

(d) Add random points R_i to the public parameters for each $i \in [1, \ell]$. Let $T_i(S) = Q_i$ if $i \in S$, and R_i otherwise. Then,

$$\begin{aligned}\mathsf{Enc}(\mathsf{pp}, \{Q_i\}, S, M) &= ([r]P, M, \gamma) \text{ where } r \xleftarrow{R} [1, n] \text{ and } \gamma = \hat{e}(T_1(S), ..., T_\ell(S))^r \\ \mathsf{Dec}_i(\mathsf{pp}, a_i, (c_1, c_2)) &= c_2\gamma_i^{-1} \text{ where } \gamma_i = \hat{e}(T_1, ..., T_{i-1}, c_1, T_{i+1}, ..., T_\ell)^{a_i}\end{aligned}$$

This encryption looks like the encryption using the scheme from (c) of a message to parties holding t_i, where $t_i = a_i$ if $i \in S$, and $t_i = r_i$ otherwise, where $R_i = [r_i]P$. If $i \in S$, then party i belongs to this set of parties. If $i \notin S$, then party i does not belong to this set of parties, since their a_i is completely independent of $t_i = r_i$. Thus, the security of this scheme reduces to the security of the scheme from (c).

2. Suppose we have an adversary $\mathcal{A}$ that wins the signature game with probability at least ϵ. That is, with probability at least ϵ, $\mathcal{A}$ outputs (m^*, σ^*) such that, if we choose a random $\tilde{m}$ and interpret m^* as an ID id^* and σ^* as a secret key sk^*, then $\mathsf{Dec}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, \mathsf{id}^*, \tilde{m}), \mathsf{sk}^*) = \tilde{m}$

   Now, we construct an algorithm $\mathcal{B}$ that breaks the underlying IBE. $\mathcal{B}$ works as follows:

   - On input pp, send $\mathsf{pk}_S = pp$ to $\mathcal{A}$.
   - When $\mathcal{A}$ asks for a signature on a message m, interpret m as an ID id. Then query the extraction oracle on id, getting back a secret key $\mathsf{sk}_{\mathsf{id}}$. Return to $\mathcal{A}$ the signature $\sigma = \mathsf{sk}_{\mathsf{id}}$.
   - When $\mathcal{A}$ produces a forgery candidate (m^*, σ^*), generate a random message $\tilde{m}$, interpret m^* as the ID id^*, and send $(\mathsf{id}^*, \tilde{m})$ as the challenge query.
   - When the challenger responds with the ciphertext c, interpret σ^* as the secret key sk^*, and check that $\mathsf{Dec}(\mathsf{pp}, c, \mathsf{sk}^*) = \tilde{m}$. If so, output 1, otherwise output 0.

   pk_S is generated the same way as it is in the standard signature game. Further, signature queries are answered the same way as well. Thus, the view of $\mathcal{A}$ as a subroutine of $\mathcal{B}$ is identical to that in the standard signature game. Suppose the challenger encrypts the message $\tilde{m}$. Then, since $\tilde{m}$ is chosen randomly, with probability at least ϵ,

$$\mathsf{Dec}(\mathsf{pp}, c, \mathsf{sk}^*) = \mathsf{Dec}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, \mathsf{id}^*, \tilde{m}), \mathsf{sk}^*) = \tilde{m}$$

This means that with probability at least ϵ, $\mathcal{B}$ outputs 1.

Now, suppose the challenger decided to encrypt a random message M. Then

$$\mathsf{Dec}(\mathsf{pp}, c, \mathsf{sk}^*) = \mathsf{Dec}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, \mathsf{id}^*, M), \mathsf{sk}^*)$$

This quantity is completely independent of $\tilde{m}$. Thus, suppose it evaluates to some message M' (it may actually fail to decrypt, but that only helps us), which depends on M, sk^*, and id^*. Then, the probability that $M' = \tilde{m}$ is $1/r$ since $\tilde{m}$ is chosen independently of M, sk^*, and id^*, and is chosen uniformly in a space of r elements. If the decryption fails, the probability is 0. Thus the probability that the decryption is equal to $\tilde{m}$ is at most $1/r$, meaning $\mathcal{B}$ outputs 1 with probability $1/r$.

If we assume $1/r < \epsilon$, the advantage of $\mathcal{B}$ is thus at least $\epsilon - 1/r$.

Bonus: We need the verification algorithm to be randomized for the last step of the proof, where we argue that the probability of the encryption of a random message decrypting to $\tilde{m}$ is at most $1/r$. If the verification algorithm used a specific message, say 0, then $\tilde{m} = 0$, and is thus not drawn from a set of size r.

To see that the proof cannot be fixed in this case, start with a secure IBE scheme $(\mathsf{Gen}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$, and construct the following modified IBE $(\mathsf{Gen}', \mathsf{Extract}', \mathsf{Enc}', \mathsf{Dec}')$:

- $\mathsf{Gen}'() = \mathsf{Gen}()$ is the same.
- $\mathsf{Extract}'(\mathsf{pp}, \mathsf{mk}, \mathsf{id}) = \mathsf{Extract}(\mathsf{pp}, \mathsf{mk}, \mathsf{id})||0$ is the same, except with the bit 0 appended to the end.
- $\mathsf{Enc}'(\mathsf{pp}, \mathsf{id}, m) = \mathsf{Enc}(\mathsf{pp}, \mathsf{id}, m)$ is the same.
- $\mathsf{Dec}'(\mathsf{pp}, c, \mathsf{sk}||b) = \mathsf{Dec}(\mathsf{pp}, c, \mathsf{sk})$ if $b = 0$, and 0 if $b = 1$. This decrypts correctly if $b = 0$, and simply outputs 0 if $b = 1$.

It is easy to see that the security and correctness of this scheme reduce to that of the underlying IBE. However, now follow the signature construction using this IBE, and suppose that our verification algorithm only tests the message 0, instead of a random message. An adversary can now sign any message m with the signature $\mathsf{sk}||1$, where sk is any random junk. Since decryption with this secret key will always produce 0, the verification algorithm will always accept.

3. (a) Recall that $\phi_{p^i} = \phi_p^i$, and that a point Q is in $E(\mathbb{F}_{p^i})$ if and only if $\phi_{p^i} Q = \phi_p^i Q = Q$. Let ϕ denote ϕ_p.
   Let $P \in E(\mathbb{F}_{p^k})$ (and thus $\phi^k P = P$), and let $Q = Tr(P)$.

$$\begin{aligned} \phi Q &= \phi Tr(P) = \phi \sum_{i=0}^{k-1} \phi^i P = \sum_{i=1}^{k} \phi^i P = \sum_{i=1}^{k-1} \phi^i P + \phi^k P \\ &= \sum_{i=1}^{k-1} \phi^i P + P = \sum_{i=0}^{k-1} \phi^i P = Tr(P) = Q \end{aligned}$$

   Thus, $Q = Tr(P) \in E(\mathbb{F}_p)$

(b) Suppose $P \in E[n]$ has eigenvalue p. Recall that $n|p^k - 1$. Then

$$\begin{aligned}(p-1)Tr(P) &= (p-1)\sum_{i=0}^{k-1}\phi^i P = (p-1)\sum_{i=0}^{k-1}p^i P \\ &= (p-1)\left(\sum_{i=0}^{k-1}p^i\right)P = (p^k-1)P = \infty\end{aligned}$$

Thus, $Tr(P)$ is a point of order dividing $p-1$. $Tr(P)$ is also a point of order dividing n (since $P \in E[n]$), and n is prime, so the order of $Tr(P)$ is either 1 or n. However, since the embedding degree of n is at least 2, $n \nmid p-1$. Hence, the order of $Tr(P)$ is 1, and hence $Tr(P) = \infty$.

(c) Let $\psi(P) = kP - Tr(P)$. Clearly, ψ is a homomorphism, so it satisfies $\psi(aQ + bR) = a\psi(Q) + b\psi(R)$. Let Q be an eigenvector of ϕ with eigenvalue 1, and R be an eigenvector with eigenvalue p. Then we can write any point $P \in E[n]$ as $aQ + bR$. Note that:

$$\begin{aligned}\psi(Q) &= kQ - Tr(Q) = kQ - \sum_{i=0}^{k-1}\phi^i Q = kQ - \sum_{i=0}^{k-1}Q = kQ - kQ = \infty \\ \psi(R) &= kR - Tr(R) = kR - \infty = kR\end{aligned}$$

Thus, $\psi(P) = a\psi(Q) + b\psi(R) = kbR$, and therefore

$$\phi\psi(P) = \phi(kbR) = kbpR = p\psi(P)$$

So as long as $\psi(P) \neq 0$, $\psi(P)$ has eigenvalue p.

(d) Let $R = aP + bQ$ where Q is an eigenvector of ϕ with value p.

$$e_n(P,R) = e_n(P, aP + bQ) = e_n(P,P)^a e_n(P,Q)^b = e_n(P,Q)^b$$

Therefore, since n is prime, $e_n(P,R)$ is a generator as long as $b \neq 0$.

$$Tr(R) = aTr(P) + bTr(Q) = akP + b\infty = akP$$

Therefore, since $n \nmid k$, $Tr(R) \neq \infty$ as long as $a \neq 0$. Since n is prime, this implies that the image of Tr over $\mathcal{G}_R$ is a group of order n, so it must be all of $\mathcal{G}_1$. Since $\mathcal{G}_R$ and $\mathcal{G}_1$ both have n elements, it must also be a bijection.

Hence, as long as $a, b \neq 0$, both conditions are met. Since R is chosen randomly, a and b are random in $[1, n]$, so the conditions are met with probability $(1 - 1/n)^2$.

4. (a) $\mathsf{Dec}(\mathsf{sk}, A, B) = \log_{[c-ab]P}([c]A - [a]B)$ where log is your favourite discrete log algorithm.

If (A, B) is an encryption of m, then $A = [m]P + [r]Q = [mb + ar]P$ and $B = [m]R + [r]S = [mb + cr]P$. Then $[c]A - [a]B = [cm + acr - mbc - acr]P = [m][c - ab]P$, so $m = \log_{[c-ab]P}([c]A - [a]B)$.

(b) Suppose we have an adversary $\mathcal{A}$ with a real-or-zero advantage at least 2ϵ. Let $\mathcal{B}_i$ for $i = 0, 1$ be the following DDH algorithms:

* On input (P, Q, R, S), run $\mathcal{A}$ with public key (P, Q, R, S).
* When $\mathcal{A}$ produces the challenge plaintext m, generate random r. If $i = 0$, return $([m]P+[r]Q, [m]R+[r]S)$ the proper encryption of m. If $i = 1$, return $([r]Q, [r]S)$, the proper encryption of 0.
* When $\mathcal{A}$ outputs a bit b, return that bit.

Let's call the public key $(P, Q, R, S) = (P, [a]P, [b]P, [c]P)$ good if $c \neq ab$ and bad if $c = ab$. Let's consider four scenarios:

* pk is good, and the challenger for $\mathcal{A}$ outputs an encryption of m. This is identical to the case where B_0 received a DDH tuple with $c \neq ab$.
* pk is good, and the challenger for A outputs an encryption of 0, that is $([r]Q, [r]S)$. This is identical to the case where B_1 received a DDH tuple with $c \neq ab$.
* pk is bad, and the challenger for A outputs an encryption of a m. This is identical to the case where B_0 received a DDH tuple with $c = ab$. Note that the ciphertext in this case is

  $$([m]P+[r]Q, [m]R+[r]S) = ([m+ar]P, [mb+rab]P) = ([m+ar]P, [b][m+ar]P)$$

  Since the order of P is a prime n, $m + ar$ is a random element mod n (as long as $a \neq n$. $a = n$ with only negligible probability, and in that case we could easily break the scheme anyway). Thus, the ciphertext is equivalent to $([r']P, [b][r']P)$ for a random r'.
* pk is bad, and the challenger for A outputs an encryption of 0. This is identical to the case where B_1 received a DDH tuple with $c = ab$. Note that the ciphertext is $([r]Q, [r]S) = ([ra]P, [rab]P) = ([ra]P, b[ra]P)$. Again, unless $a = 0$, ra is a random element mod n, so the ciphertext is equivalent to $([r']P, [b][r']P)$ for random r'. Therefore, B_0 and B_1 behave identically if $c = ab$.

We can now present a hybrid argument:

$$\begin{aligned}
2\epsilon &< Adv(\mathcal{A}) = |\Pr[A = 1|Enc(m),\ \text{good } \mathsf{pk}] - \Pr[A = 1|Enc(0),\ \text{good } \mathsf{pk}]| \\
&< |\Pr[A = 1|Enc(m),\ \text{good } \mathsf{pk}] - \Pr[A = 1|Enc(m),\ \text{bad } \mathsf{pk}]| \\
&\quad + |\Pr[A = 1|Enc(m),\ \text{bad } \mathsf{pk}] - \Pr[A = 1|Enc(0),\ \text{good } \mathsf{pk}]| \\
&= |\Pr[A = 1|Enc(m),\ \text{good } \mathsf{pk}] - \Pr[A = 1|Enc(m),\ \text{bad } \mathsf{pk}]| \\
&\quad + |\Pr[A = 1|Enc(0),\ \text{bad } \mathsf{pk}] - \Pr[A = 1|Enc(0),\ \text{good } \mathsf{pk}]| \\
&= Adv(B_0) + Adv(B_1)
\end{aligned}$$

Therefore, at least one of $Adv(B_0)$ and $Adv(B_1)$ is greater that ϵ, a contradiction to our assumption that DDH is hard.

(c) $Add((A_1, B_1), (A_2, B_2)) = (A_1 + A_2 + [r]Q, B_1 + B_2 + [r]S)$ for a randomly generated r. If $(A_1, B_1) = ([m_1]P + [r_1]Q, [m_1]R + [r_1]S)$ and $(A_2, B_2) = ([m_2]P + [r_2]Q, [m_2]R + [r_2]S)$, then adding the two encryption gives:

$$([m_1 + m_2]P + [r_1 + r_2 + r]Q, [m_1 + m_2]R + [r_1 + r_2 + r]S)$$

This is the encryption of $m_1 + m_2$ with randomness $r' = r_1 + r_2 + r$. Since r is generated randomly, r' is random, and independent of r_1 and r_2, as desired.

(d) Suppose

$$\begin{aligned}(A, B) &= ([m_1]P + [r_1]Q, [m_1]R + [r_1]S) = ([m_1 + ar_1]P, [bm_1 + cr_1]P)\\ (C, D) &= ([m_2]P' + [r_2]Q', [m_2]R' + [r_2]S') = ([m_2 + a'r_2]P', [b'm_2 + c'r_2]P')\end{aligned}$$

For notational convenience, let

$$\begin{aligned}s &= e_n(P, P')^{m_1 m_2}\\ t &= e_n(P, P')^{r_1 m_2}\\ u &= e_n(P, P')^{m_1 r_2}\\ v &= e_n(P, P')^{r_1 r_2}\end{aligned}$$

Notice that

$$\begin{aligned}w &= e_n(A, C) = e_n(P, P')^{(m_1 + ar_1)(m_2 + a'r_2)} = st^a u^{a'} v^{aa'}\\ x &= e_n(A, D) = e_n(P, P')^{(m_1 + ar_1)(b'm_2 + c'r_2)} = s^{b'} t^{ab'} u^{c'} v^{ac'}\\ y &= e_n(B, C) = e_n(P, P')^{(bm_1 + cr_1)(m_2 + a'r_2)} = s^b t^c u^{a'b} v^{a'c}\\ z &= e_n(B, D) = e_n(P, P')^{(bm_1 + cr_1)(b'm_2 + c'r_2)} = s^{bb'} t^{b'c} u^{bc'} v^{cc'}\end{aligned}$$

Our goal now is to eliminate t, u, v (which are unknown) to recover s. This is actually a linear algebra problem, where s, t, u, v are unknown vectors, and we know specific linear combinations of them (where vector addition is multiplication, and scalar multiplication is exponentiation). Our goal is to, knowing the coefficients in the linear combination, recover the vectors. Using standard techniques, we can get that

$$e_n(P, P')^{m_1 m_2} = s = w^{\frac{cc'}{(c-ab)(c'-a'b')}} x^{\frac{-a'c}{(c-ab)(c'-a'b')}} y^{\frac{-ac'}{(c-ab)(c'-a'b')}} z^{\frac{aa'}{(c-ab)(c'-a'b')}}$$

Where the arithmetic in the exponents is carried out mod n.

5. (a) We know that the only possibilities for $\#E(\mathbb{F}_{2^d})$ are $2^d + 1 - t$ with $t = 0, \pm\sqrt{2^d}$(if d is even),$\pm\sqrt{2 \times 2^d}$(if d is odd). In the first and second case, we know the embedding degree is at most 2 and 1, respectively. Therefore, we will need to look at the third case. We will need to choose an odd d such that one of $2^d + 1 \pm \sqrt{2^{d+1}}$ has a large prime factor $r > 2^{160}$ (which will mean there is a subgroup of size r), and for which the embedding degree of r is 4. This last condition is equivalent to

$r \nmid 2^d - 1$, $r \nmid 2^{2d-1} - 1 = (2^d - 1)(2^d + 1)$, and $r \nmid 2^{3d} - 1 = (2^d - 1)(2^{2d} + 2^d + 1)$. Since r is prime, this is equivalent to checking if $r \nmid (2^{3d} - 1)(2^d + 1)$.

We also need $2^{4d} > 2^{1000}$, or $d > 250$, for the output of the Weil pairing to not take values in any field of size $\leq 2^{1000}$. Once we've found d and r, we just need to find a curve with the specified number of elements. Therefore, we get the following algorithm:

```
def find_curve(mind_d,min_r,base,const):
    for d in range(min_d,1000,2):
        n = Integer(base^d+1-base^((d+1)/2))
        r = max(factor(n))[0]
        if r > min_r:
           F=GF(base^d,'a')
           for A in [1,base-1]:
              E=EllipticCurve(F,[0,0,1,A,const])
              if E.order() % r == 0:
                  return (d,r,A)
```

Notice that we only look for $A \in \mathbb{F}_(2)$ and only check the minus solution. Running this algorithm for base 2 with the minimum values specified, we get

```
min_d = 251
min_r = 2^160
base = 2
const = 0
find_curve(min_d,min_r,base,0)
```

```
(253, 621109541542884571802304568790331501283098925929529, 1)
```

Therefore, with $d = 253$, and $A = 1$, we get a curve with a subgroup of order 621109541542884571802304568790331501283098925929529. Now we just need to check that $r \nmid (2^{3d} - 1)(2^d + 1)$:

```
(2^(3*253)-1)*(2^253-1) % 621109541542884571802304568790331501283098925929529
```

```
2
```

(b) Here, we have the same reasoning, except that now we are looking for a curve with $3^d + 1 \pm \sqrt{3^{d+1}}$. Since we want the embedding degree of r to be 6, we must check that $r \nmid 3^d - 1$, $r \nmid 3^{2d} - 1 = (3^d - 1)(3^d + 1)$, $r \nmid 3^{3d} - 1 = (3^d - 1)(3^{2d} + 3^d + 1)$, $r \nmid 3^{4d} - 1 = (3^d - 1)(3^d + 1)(3^{2d} + 1)$, and $r \nmid 3^{5d} - 1 = (3^d - 1)(3^{4d} + 3^{3d} + 3^{2d} + 3^d + 1)$. This is equivalent to $r \nmid (3^{5d} - 1)(3^{2d} + 3^d + 1)(3^{2d} + 1)(3^d + 1)$. Now, rather than have $4d > 1000$, we only need $6d > 1000$, so $d > 166$:

```
min_d = 167
min_r = 2^160
base = 3
const = 1
find_curve(min_d,min_r,base,1)
```

```
  ***   Warning: Mod(a,b)^n with n >> b : wasteful.
(167,
135178432469278085190543632884974507130517958441098016641911005557482339\
, 2)
```

Therefore, with $d = 167$, $A = 2$, we get a curve with a subgroup of order 135178432469278085190543632884974507130517958441098016641911005557482339. Now we just need to check that $r \nmid (3^{5d} - 1)(3^{2d} + 3^d + 1)(3^{2d} + 1)(3^d + 1)$:

```
(3^(5*167)-1)*(3^(2*167)+3^167+1)*(3^(2*167)+1)*(3^167+1) %
135178432469278085190543632884974507130517958441098016641911005557482339
```

```
239450303651240395772054800534340942113 62
```

(c) Suppose $p(\alpha) > 25$, meaning $p(\alpha) > 4\sqrt{p(\alpha)}+1$. Then since $r(\alpha) = p(\alpha)-6\alpha^2 > p(\alpha)-\sqrt{p(\alpha)}$, we have that $2r(\alpha) > 2p(\alpha)-2\sqrt{p(\alpha)} > p(\alpha)+2\sqrt{p}(\alpha)+1$, meaning $2r(\alpha)$ is not a possibility for the number of points on an elliptic curve over $\mathbb{F}_{p(\alpha)}$ (an similarly, no higher multiple is either). Therefore, to test if the order of $E(\mathbb{F}_{p(\alpha)})$ is $r(\alpha)$, we only need to check if a non-infinity element has order $r(\alpha)$, or equivalently, if $[r(\alpha)]P = \infty$. This gives us the following algorithm:

```
def find_curve2(min_a):
    for i in range(10000):
        a=i+min_a
        p = 36*a^4+36*a^3+24*a^2+6*a+1
        r = p-6*a^2
        if is_prime(r) and is_prime(p):
            F = GF(p)
            for B in range(1,100):
                E=EllipticCurve(F,[0,B])
                P = E.random_element()
                if r*P == E(0):
                    return (a,B)
```

```
find_curve2(2^63)
```

```
(9223372036854776665, 12)
```

Thus, $\alpha = 9223372036854776665$ gives us prime $r(\alpha)$ and $p(\alpha)$, and $B = 12$ gives us a curve with order $r(\alpha)$. To check that the embedding degree is 12:

```
a = 9223372036854776665
p = 36*a^4+36*a^3+24*a^2+6*a+1
r = p-6*a^2
for i in [1,12]:
    if (p^i-1) % r == 0:
        print i
```

```
12
```