j2tham

1a) 1. Uses python random module which is only pseudorandom. I would use the secrets or sympy modules instead

2. Cipher block mode is ECB, which is less secure than other styles such as CBC. I would use CBC

3. provides n, e, c_1 and c_2 in the JSON file. I would encrypt n,e,c_1 in a second file and send it to myself

4. e was not selected specifically for RSA, (where it should be $1 < e < \phi(n)$ and that $\gcd(e, \phi(n)) = 1$). I would ensure that e is chosen within these specs and reselect e if necessary

1b)

```
# TODO
phi = totient(n)
d = mod_inverse(e,phi)
aes_key_int = pow(c_1,d,n)
aes_key = aes_key_int.to_bytes((aes_key_int.bit_length()+7)//8,byteorder='big')
cipher = Cipher(algorithms.AES(aes_key),modes.ECB())
decryptor = cipher.decryptor()
padded = decryptor.update(c_2)+decryptor.finalize()
unpadder = padding.PKCS7(128).unpadder()
plaintext = unpadder.update(padded)+unpadder.finalize()
# write the decrypted assignment to a file
with open("assignment_out.pdf", 'wb') as fh:
    fh.write(plaintext)
```

1c) 96106

2)

Diffie-Hellman assumption (DHA) - given $g, g^a \; and \; g^b$, it is computationally infeasible to determine g^{ab}.

discrete logarithm assumption (DLA)- given $g \; and \; g^a$, it is computationally infeasible to determine a

In a scenario where DLA does not hold, we can trivially break DHA in the following ways:

Given A' where g^{a^2} can be calculated with g and g^a, a is determined. g^{a^2} can be calculated by $a * g^a$. DHA is also broken as g^{ab} can be calculated efficiently where g^{ab} is calculated by $g^a * g^b$. Since DLA is similar to DHA, given A' that can calculate a or b given g^a or g^b respectively, DHA is trivial broken. Hence, the square DHA would not hold either. Thus, by contrapostive, square DHA is equivalent to DHA

3a) For public keys $X = g^x \ and \ Y = g^y$, make a call to O_D under XY which would return m. Given $(c_0, c_1) = (g^r, m(g^{xy})^r)$, we can calculate g^{xy} by taking $\frac{c_1}{c_0 * m}$.

3b) Input the values $X = Z = g^z, Y = c_0 = g'$ into O_{DH}. By querying this oracle, we obtain g^{zr}. Since $c_1 = m(g^z)^r$, we can compute $\frac{m(g)^{zr}}{g^{zr}}$ to obtain m.

4 a) Looking at the power consumption graph, we can infer the value 1010110101111. Thus, the most significant byte in Alice's private key corresponds to 10101101.

b) $P = (2, 3), Q = (5, 2),\ y^2 = x^3 - x + 3$

i. $m = \frac{3*2^2-1}{2*3} = \frac{11}{6} = \frac{4}{6} = \frac{2}{3} = \frac{2}{10} = \frac{1}{5} = 3,$

$$x_{P+P} = m^2 - x_P - x_P = 3^2 - 2 - 2 = 5,$$

$$y_{P+P} = -(m(x_{P+P} - x_P) + y_P) = -(3 * (5 - 2) + 3) = -12 = -5 = 2,$$

$P + P = (5, 2)$

ii. $m = \frac{2-3}{5-2} = -\frac{1}{3} = 2,$

$$x_{P+Q} = m^2 - x_P - x_q = 2^2 - 2 - 5 = -3 = 4,$$

$$y_{P+Q} = -\left(m\left(x_{P+Q} - x_P\right) + y_P\right) = -(2 * (4 - 2) + 3) = -7 = 0,$$

$$P + Q = (4, 0)$$

iii. $m = \frac{3*5^2-1}{2*2} = \frac{37}{2} = \frac{2}{2} = 1$

$$x_{Q+Q} = m^2 - x_q - x_q = 1^2 - 5 - 5 = -9 = -2 = 5,$$

$$y_{Q+Q} = -\left(m\left(x_{Q+Q} - x_Q\right) + y_Q\right) = -(1 * (5 - 5) + 2) = -2 = 5,$$

$$Q + Q = (5, 5)$$

c) $using\ m = \frac{\left(x_P+x_Q\right)^2 - x_P x_Q + a}{y_P + y_Q},$

iv. $for\ P + P,\ m = \frac{(2+2)^2-2*2-1}{3+3} = \frac{11}{6} = 3,\ P + P = (5, 2)$

v. $for\ P + Q,\ m = \frac{(2+5)^2-2*5-1}{2+3} = \frac{38}{5} = \frac{3}{5} = \frac{3}{12} = \frac{1}{4} = 2$

$$x_{P+Q} = 2^2 - x_P - x_q = 2^2 - 2 - 5 = -3 = 4$$

vi. $for\ Q + Q,\ m = \frac{(5+5)^2-5*5-1}{2+2} = \frac{37}{2} = 1,\ Q + Q = (4, 0)$

d) Add noise to the emitted channel by introducing arbitrary and artificial noise via random delays.

5a) If $k = 0$, s will be undefined.

5b) To verify the signature as valid for DSA signing, we check for $0 < r < q,\ and\ 0 < s < q,\ and\ \left(g^{\frac{H(m)}{s}} g^{\frac{\propto r}{s}}\ mod\ p\right)\ mod\ q = r$. For $r = 0$, we check for $\left(g^{\frac{H(m)}{s}}\ mod\ p\right)\ mod\ q = 0$. $s = \frac{H(m)}{k}\ mod\ q$, thus, we have $\left(g^{\frac{H(m)}{\frac{H(m)}{k}}}\ mod\ p\right)\ mod\ q = (g^k\ mod\ p)\ mod\ q = 0$. Since $r = (g^k\ mod\ p) = 0$, we have $0\ mod\ q = 0$, and this statement will always hold for any value of s and thus the attacker can forge a signature on any message

5c) For $s = 0,$ we check for $(\infty\ mod\ p)\ mod\ q = r$. Since ∞ is unable to be calculated, it is no longer required to verify as valid, and any value of r will be valid allowing the attacker to forge a signature.