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 and that ). 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 it is computationally infeasible to determine

discrete logarithm assumption (DLA)- given it is computationally infeasible to determine

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

Given A’ where can be calculated with and , a is determined. can be calculated by . DHA is also broken as can be calculated efficiently where is calculated by . Since DLA is similar to DHA, given A’ that can calculate a or b given or 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 , make a call to under which would return . Given , we can calculate by taking .

3b) Input the values into . By querying this oracle, we obtain . Since , we can compute to obtain .

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

b)

* + 1. ,

* + 1. ,

c)

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

5a) If will be undefined.

5b) To verify the signature as valid for DSA signing, we check for For , we check for . thus, we have . Since we have , and this statement will always hold for any value of and thus the attacker can forge a signature on any message

Since is unable to be calculated, it is no longer required to verify as valid, and any value of will be valid allowing the attacker to forge a signature.

Discussed with Sean