Assignment 2: Public Key Cryptography and Hash Functions
Cryptography and Secure Communications (MITS5500)
Prepared by: Miguel V. Martin, PhD, PEng
Due: October 30th, 2020 by 11:59 pm

This is an individual assignment; collaboration is not allowed. Don't zip your submission.

1. RSA [Marks: 25%] In a public-key system using RSA, you intercept the ciphertext $C = 10$ sent to a user whose public key is $\{e = 5, n = 35\}$. Show all your work (either Math or Python) to answer the following two questions (not necessarily in that order):
a) What is the plaintext *M*? Hint: with $n = 35$, a brute force may save you headaches.
b) What is the secret exponent *d* in the private key $\{d = ?, n = 35\}$?

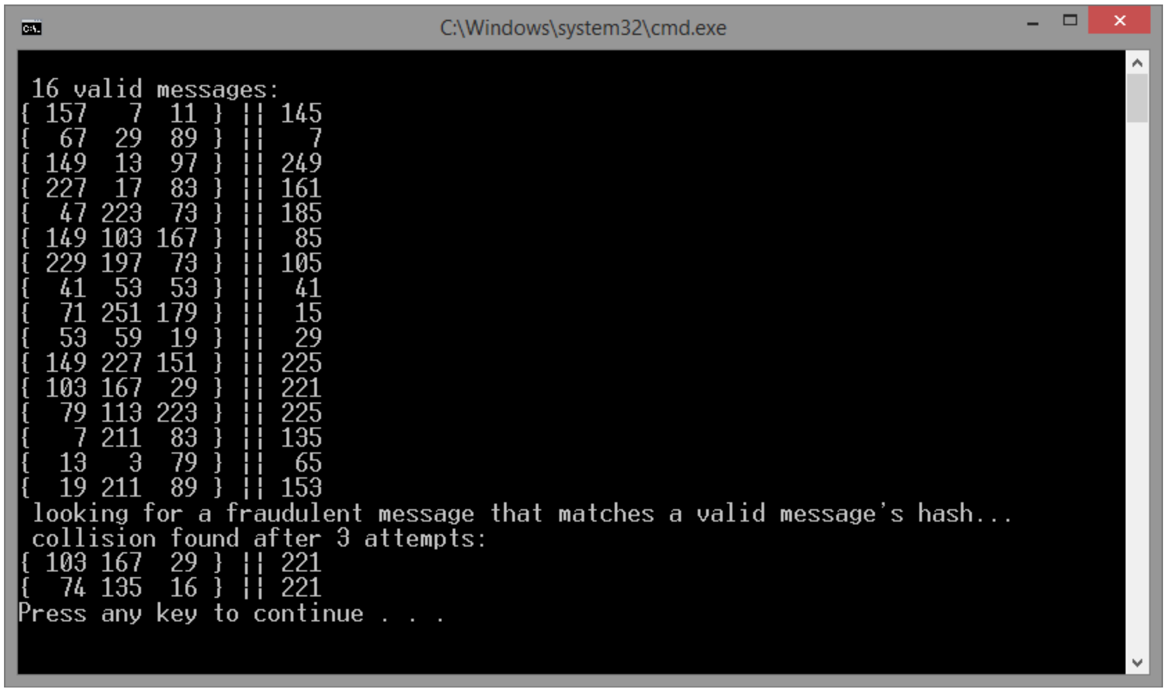
2. Key Exchange [Marks: 25%]. In a Diffie-Hellman scheme with a common prime $q = 11$ and a primitive root $\alpha = 2$:
a) If user *A* has public key $Y_A = 9$, what is *A*'s private key *X*?
b) If user *B* has public key $Y_B = 3$, what is the secret key *K*, shared with *A*?
Show all your work (either Math or Python).

3. The Birthday Paradox Attack [Marks 25%]. Johnny, an Ontario Tech graduate recently got a job at CSE in Ottawa. As part of a pilot project, every month CSE announces a list of 3 prime numbers (all less than 2^8) in proxy repositories which automatically sign the 3 numbers for Navy bases across the Country to fetch them. These numbers become part of public keys so confidentiality is not required, only signature. In his very first day at work, Johnny volunteers to take charge of the automatic signature generation in the Bay of Fundy proxy (as it was close enough to his native Saint John). He decided to sign the 3 numbers using a hash value consisting of the bitwise XOR amongst the bit representation of these 3 numbers and then encrypting the hash value using CSE's private RSA key. Further, he made sure his automatic process only signed the numbers if all of the 3 numbers were prime (i.e., if at least one of the 3 numbers were composite the process would refuse to sign).

Last month's numbers were `{251, 157, 191}`. Johnny's program computed the hash to be `217` (because 251 XOR 157 XOR 191 = 217) which then was encrypted with RSA and appended to the set of numbers as signature. Shortly after, CSE suspected a probable compromise of the Bay of Fundy proxy since the Nova Scotia Navy Base ended up using composite numbers in their crypto operations causing serious disruptions in the pilot project! During his debrief, Johnny claimed that to forge the signature with probability ½, an attacker would need to compute the hash values of $\frac{2^8}{2} = 2^7 = 128$ sets of 3 random composite numbers, not to mention that the attacker would require to break RSA. But his boss, an experienced cryptographer advises that Johnny's claims were overly optimistic, saying that the attacker could forge a signature with probability greater than ½ by computing the hashes of only $2^{8/2} = 2^4 = 16$ (not 128) sets of 3 random composite numbers, and that the attacker wouldn't even have to break RSA at all. Johnny was speechless! In her explanation, Johnny's boss alluded to the birthday paradox, a concept that for some reason Johnny never quite grasped while taking MITS5500 at Ontario Tech. Help Johnny understand the birthday paradox, after the fact, by following these steps:

(a) Compute the hashes of $2^4 = 16$ sets of 3 random prime numbers less than 2^8.
(b) Then keep selecting sets of 3 fraudulent composite numbers less than 2^8 until you find a set `{c₁, c₂, c₃}` with a hash value that matches one of the hashes computed in step (a); we will refer to that matching set as `{p₁, p₂, p₃}`.
(c) Explain to Johnny that the attacker would replace the legitimate prime set `{251, 157, 191}` with the fraudulent composite set `{c₁, c₂, c₃}` in the Bay of Fundy proxy server. The attacker would then use Johnny's automatic signing process to sign the prime set `{p₁, p₂, p₃}` and append such signature to the composite set `{c₁, c₂, c₃}`. At this point the attacker has been successful at signing the composite set `{c₁, c₂, c₃}` without knowing CSE's RSA private key and by computing just over 16 hashes (i.e., 16 from Part (a) + *some* from Part (b)).
(d) Submit a screenshot like the one depicted in the figure below.

(Notes: Of course, Johnny wasn't so naïve as to suggest such a simple scheme for signing; in reality he was using SHA-256 but here we are simplifying things just to get the concept across (see Question 4). Also, CSE uses prime numbers in the order of 2^{2048}, not 2^8. As per Johnny's future at CSE, don't worry; his boss decided to give him a second chance. Johnny was spotted last evening close to the ByWard Market enjoying a beaver tail with his partner.)


C:\Windows\system32\cmd.exe

```
 16 valid messages:
{ 157   7  11 } || 145
{  67  29  89 } ||   7
{ 149  13  97 } || 249
{ 227  17  83 } || 161
{  47 223  73 } || 185
{ 149 103 167 } ||  85
{ 229 197  73 } || 105
{  41  53  53 } ||  41
{  71 251 179 } ||  15
{  53  59  19 } ||  29
{ 149 227 151 } || 225
{ 103 167  29 } || 221
{  79 113 223 } || 225
{   7 211  83 } || 135
{  13   3  79 } ||  65
{  19 211  89 } || 153
 looking for a fraudulent message that matches a valid message's hash...
 collision found after 3 attempts:
{ 103 167  29 } || 221
{  74 135  16 } || 221
Press any key to continue . . .
```

4. Attempting a Birthday Attack on SHA-256 [Marks: 25%]. Write a Python program using cryptography.io that tries to find a collision like in Question 3 but now instead of XOR-ing the three prime numbers, use SHA-256 as in the Python sample code below (from https://cryptography.io/en/latest/hazmat/primitives/cryptographic-hashes/). Let your program generate a thousand hash values of random fraudulent messages (i.e., sets of three composite numbers) and report whether a collision was found as in Question 3. Submit a screenshot like the one depicted in the figure above for Question 3 (but don't show the 1,000 values; 10 would do). <u>Quite likely, you won't find a collision; but if you do, you will get full marks in this assignment regardless of the marks in the other questions</u>. Oh! And don't worry about Johnny, he has decided to go back to Ontario Tech for a Doctoral degree under the supervision of Prof. Martin, while yet declining a promotion at CSE.

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
digest = hashes.Hash(hashes.SHA256(), backend=default_backend())
digest.update(b"251")
digest.update(b"157")
digest.update(b"191")
digest.finalize()
```