Partners: Ntense Obono and Ugo Anyaegbunam

HERE IS DIFFIE HELLMAN SECTION
Figure out the shared secret agreed upon by Alice and Bob. This will be an integer.

- The shared secret is 65

Show your work. Exactly how did you figure out the shared secret?

1. We are given the g and p agreed on by Alice and Bob, We are also given the number Alice sent to Bob and the number that Bob sent to Alice
2. Since we know that the shared secret will be the same for both Alice and Bob, we then found the secret number that Alice selected
3. The code below was computed to find the secret key and then compute the shared secret with all the information we had.
4. We then verified that the shared secret for alice was the same as that of bob

```
g, p = 7, 97
A, B = 53, 82

def find_key(person, g=g, p=p):
   '''
   This Function finds the secret number selected by a person [Alice or Bob] given g
and p
   and returns the secret number
   '''
   x = 1
   while True:
       if (g**x)%p == person:
           return x
       else:
           x += 1
ax = find_key(A)
bx = find_key(B)

alice_shared_secret = (B**ax) % p
bob_shared_secret = (A**bx) % p

if shared_secret == shared_secret_2:
   print(f"The shared secret is {alice_shared_secret}")
else:
   print("You messed up your calculations")
```

Show precisely where in your process you would have failed if the integers involved were much larger.

Partners: Ntense Obono and Ugo Anyaegbunam

Below is the snippet of code that would have failed if the integers involved were much larger. The reasoning for this is because we are using a brute force algorithm and we would get stuck in the while loop potentially crashing our computer the higher the integers get

```
while True:
    if (g**x)%p == person:
        return x
    else:
        x += 1
```

HERE IS RSA SECTION
HERE IS HOW WE DECRYPTED ALICE MESSAGE
>>> e = 13
>>> n = 162991
//This would have gotten pretty nasty if n was even bigger, like a lot bigger. It could take who knows how long to hit that first factor, let alone the second.
>>> for num in range(1,n):
... if n%num == 0:
... print(num)
...
1
389
419
>>> p = 389
>>> q = 419
>>> from math import lcm
>>> lmb = lcm(p-1,q-1)
>>> lmb
81092
//This also would have been disgusting if the numbers were even bigger.
>>> for num in range(lmb):
... if num*e % lmb == 1:
... print(num)
...
43665
>>> d = 43665
>>> encrypted = [17645, 100861, 96754, 160977, 120780, 90338, 130962, 74096,
... 128123, 25052, 119569, 39404, 6697, 82550, 126667, 151824,
... 80067, 75272, 72641, 43884, 5579, 29857, 33449, 46274,
... 59283, 109287, 22623, 84902, 6161, 109039, 75094, 56614,
... 13649, 120780, 133707, 66992, 128221]
>>> decrypted = list()
>>> for num in encrypted:
... decrypted.append(num**d % n)
...

Partners: Ntense Obono and Ugo Anyaegbunam

```
>>> decrypted
[17509, 24946, 8258, 28514, 11296, 25448, 25955, 27424, 29800, 26995, 8303, 30068, 11808,
26740, 29808, 29498, 12079, 30583, 30510, 29557, 29302, 25961, 27756, 24942, 25445,
30561, 29795, 26670, 26991, 12064, 21349, 25888, 31073, 11296, 16748, 26979, 25902]
>>> for i in range(len(decrypted)):
...     decrypted[i] = hex(decrypted[i])
...
>>> decrypted
['0x4465', '0x6172', '0x2042', '0x6f62', '0x2c20', '0x6368', '0x6563', '0x6b20', '0x7468', '0x6973',
'0x206f', '0x7574', '0x2e20', '0x6874', '0x7470', '0x733a', '0x2f2f', '0x7777', '0x772e', '0x7375',
'0x7276', '0x6569', '0x6c6c', '0x616e', '0x6365', '0x7761', '0x7463', '0x682e', '0x696f', '0x2f20',
'0x5365', '0x6520', '0x7961', '0x2c20', '0x416c', '0x6963', '0x652e']
>>> for i in range(len(decrypted)):
...     decrypted[i] = decrypted[i][2:]
...
>>> decrypted
['4465', '6172', '2042', '6f62', '2c20', '6368', '6563', '6b20', '7468', '6973', '206f', '7574', '2e20',
'6874', '7470', '733a', '2f2f', '7777', '772e', '7375', '7276', '6569', '6c6c', '616e', '6365', '7761',
'7463', '682e', '696f', '2f20', '5365', '6520', '7961', '2c20', '416c', '6963', '652e']
>>> for i in range(len(decrypted)):
...     byte_string = bytes.fromhex(decrypted[i])
...     decrypted[i] = byte_string.decode("ASCII")
...
>>> decrypted
['De', 'ar', ' B', 'ob', ', ', 'ch', 'ec', 'k ', 'th', 'is', ' o', 'ut', '. ', 'ht', 'tp', 's:', '//', 'ww', 'w.', 'su', 'rv', 'ei', 'll',
'an', 'ce', 'wa', 'tc', 'h.', 'io', '/ ', 'Se', 'e ', 'ya', ', ', 'Al', 'ic', 'e.']
>>> ''.join(decrypted)
'Dear Bob, check this out. https://www.surveillancewatch.io/ See ya, Alice.'
HERE IS HOW ALICE ENCODED HER MESSAGE
>>> message = ''.join(decrypted)
>>> message
'Dear Bob, check this out. https://www.surveillancewatch.io/ See ya, Alice.'
>>> ascii_message = list()
>>> for char in message:
...     ascii_message.append(ord(char))
...
>>> ascii_message
[68, 101, 97, 114, 32, 66, 111, 98, 44, 32, 99, 104, 101, 99, 107, 32, 116, 104, 105, 115, 32, 111,
117, 116, 46, 32, 104, 116, 116, 112, 115, 58, 47, 47, 119, 119, 119, 46, 115, 117, 114, 118, 101,
105, 108, 108, 97, 110, 99, 101, 119, 97, 116, 99, 104, 46, 105, 111, 47, 32, 83, 101, 101, 32,
121, 97, 44, 32, 65, 108, 105, 99, 101, 46]
>>> hexcoded = list()
>>> for num in ascii_message:
...     hexcoded.append(hex(num))
```

Partners: Ntense Obono and Ugo Anyaegbunam

```
...
>>> hexcoded
['0x44', '0x65', '0x61', '0x72', '0x20', '0x42', '0x6f', '0x62', '0x2c', '0x20', '0x63', '0x68', '0x65',
'0x63', '0x6b', '0x20', '0x74', '0x68', '0x69', '0x73', '0x20', '0x6f', '0x75', '0x74', '0x2e', '0x20',
'0x68', '0x74', '0x74', '0x70', '0x73', '0x3a', '0x2f', '0x2f', '0x77', '0x77', '0x77', '0x2e', '0x73',
'0x75', '0x72', '0x76', '0x65', '0x69', '0x6c', '0x6c', '0x61', '0x6e', '0x63', '0x65', '0x77', '0x61',
'0x74', '0x63', '0x68', '0x2e', '0x69', '0x6f', '0x2f', '0x20', '0x53', '0x65', '0x65', '0x20', '0x79',
'0x61', '0x2c', '0x20', '0x41', '0x6c', '0x69', '0x63', '0x65', '0x2e']
>>> hexcoded_pairs = list()
>>> for i in range(0,len(hexcoded),2):
...     hexcoded_pairs.append(hexcoded[i] + hexcoded[i+1][2:])
...
>>> hexcoded_pairs
['0x4465', '0x6172', '0x2042', '0x6f62', '0x2c20', '0x6368', '0x6563', '0x6b20', '0x7468', '0x6973',
'0x206f', '0x7574', '0x2e20', '0x6874', '0x7470', '0x733a', '0x2f2f', '0x7777', '0x772e', '0x7375',
'0x7276', '0x6569', '0x6c6c', '0x616e', '0x6365', '0x7761', '0x7463', '0x682e', '0x696f', '0x2f20',
'0x5365', '0x6520', '0x7961', '0x2c20', '0x416c', '0x6963', '0x652e']
>>> og_encryption = list()
>>> for num in hexcoded_pairs:
...     og_encryption.append(int(num[2:],16))
...
>>> og_encryption
[4465, 6172, 2042, 17509, 24946, 8258, 28514, 11296, 25448, 25955, 27424, 29800, 26995,
8303, 30068, 11808, 26740, 29808, 29498, 12079, 30583, 30510, 29557, 29302, 25961, 27756,
24942, 25445, 30561, 29795, 26670, 26991, 12064, 21349, 25888, 31073, 11296, 16748,
26979, 25902]
>>> from ast import literal_eval
>>> og_encryption = list()
>>> for num in hexcoded_pairs:
...     og_encryption.append(literal_eval(num))
...
>>> og_encryption
[17509, 24946, 8258, 28514, 11296, 25448, 25955, 27424, 29800, 26995, 8303, 30068, 11808,
26740, 29808, 29498, 12079, 30583, 30510, 29557, 29302, 25961, 27756, 24942, 25445,
30561, 29795, 26670, 26991, 12064, 21349, 25888, 31073, 11296, 16748, 26979, 25902]
```

Encoding is insecure independent of RSA because there is no security around the means of decoding it. It kinda just adds an extra layer of stuff do rather than security, because anyone can decode hexadecimal by converting it. Nothing about that is anonymous or keeps people away from the decoding of it. The hex encoding just made it tedious but no more secure than it was with RSA.