Osareme Davis

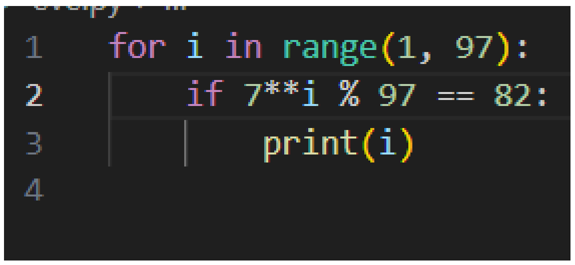
28 September 2024

Computer Security

Professor Jeff Ondich

Diffie Hellman

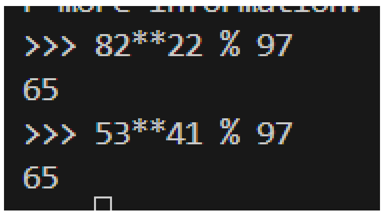
Alice and Bob's shared secret is 65. I figured this out by hunting for their chosen integers. I checked each possibility using this for loop (switching out the condition to match either Alice or Bob's information):



```
for i in range(1, 97):
    if 7**i % 97 == 82:
        print(i)
```

An issue with this method is that it has to loop a number of times equal to p. If p were significantly larger, then the worst case performance would make this method highly impractical. Larger integers also slow down the performance of operations like the exponent (at least for Python running on my laptop).

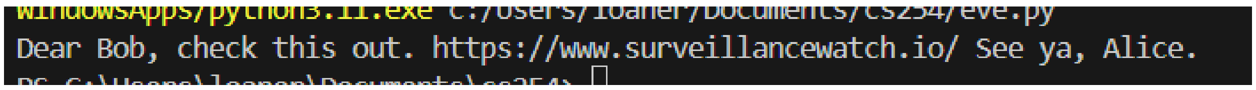
Then, once I had their chosen integers, I could calculate the shared secret (I did both Alice and Bob's calculations to double check):



```
>>> 82**22 % 97
65
>>> 53**41 % 97
65
```

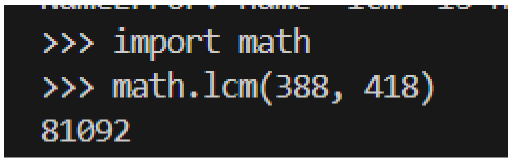
RSA

Here is Alice's message decoded:



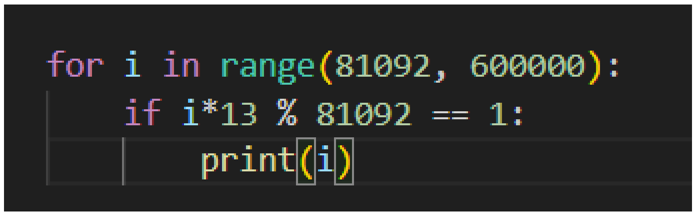
```
WindowsApps/python3.11.exe c:/Users/loaner/Documents/cs254/eve.py
Dear Bob, check this out. https://www.surveillancewatch.io/ See ya, Alice.
```

I used an online prime factor calculator to find p and q. This part seems the most problematic for larger values (finding the primes seems computationally expensive). Then calculated lambda n with the equation like normal:



```
>>> import math
>>> math.lcm(388, 418)
81092
```

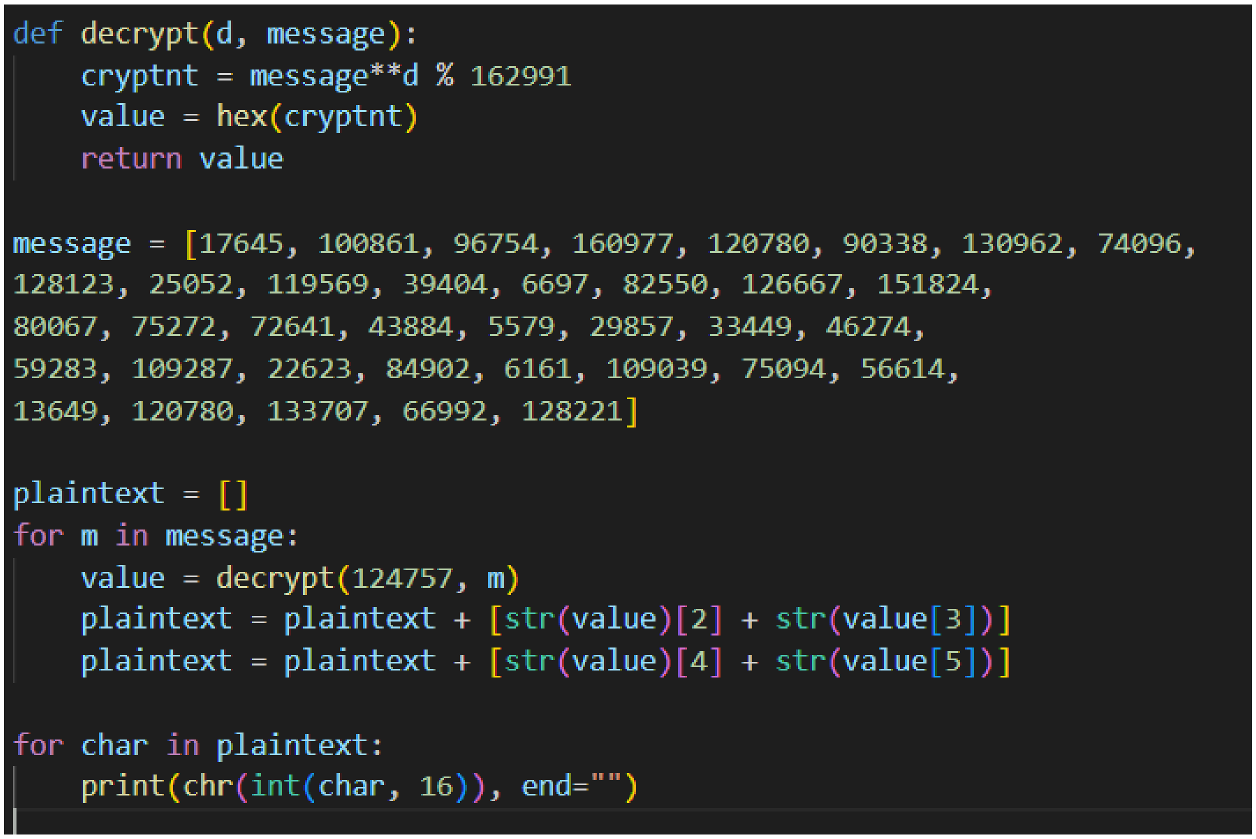
From there I guessed on d for a few values and hoped one would work:



```
for i in range(81092, 600000):
    if i*13 % 81092 == 1:
        print(i)
```

If the integers were larger and I had to guess more numbers I think I would just steal Bob's computer instead.

I tried out decrypting the first integer, then tested out a few decoding methods to see if I could check the value for d. From here I made an educated guess that Alice's message was encoded in hexadecimal and ran the decrypted data through hex(). Trying out a few characters seemed to yield pairs of the hexadecimal character encodings, so I wrote up some code to help speed up the process:



```python
def decrypt(d, message):
    cryptnt = message**d % 162991
    value = hex(cryptnt)
    return value

message = [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]

plaintext = []
for m in message:
    value = decrypt(124757, m)
    plaintext = plaintext + [str(value)[2] + str(value[3])]
    plaintext = plaintext + [str(value)[4] + str(value[5])]

for char in plaintext:
    print(chr(int(char, 16)), end="")
```

Essentially, it just grabs the pairs of hexadecimal digits and converts them into characters using the ASCII standard. I got "lucky" and this gave me the very readable decrypted plaintext message. Alice's encoding made it relatively easy to guess my way through the end of this process.