Lab 1: Exploring Symmetric Key Cryptography

Report by Ryan Blair - 9:30am section

Task III. Encrypt an Image Using OTP

View the images. *What do you observe? Are you able to derive any useful information about from either of the encrypted images? What are the causes for what you observe?*

A: It looks like the old TV screens when there was no signal with all the white noise that accompanied it. There is no useful information that can be seen from either encrypted image. The reason why is because the pixel was XOR with some random value the same size as the image, so the pixels have changed based upon the randomness of the random value calculated.

(Code - OTP)

```
rand = os.urandom(len(pic))
result = ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(pic, rand))
```

Task IV. The Two-Time Pad

View the output. *Are you able to now derive any useful information about the original plaintexts from the resulting image? What are the causes for what you observe?*

A: There is useful information in the form of the 2 images have been transposed onto one another. Anytime the logos are interacting with the other's white background, the colors of the logo have been inverted due to the XOR calculation with white making it the opposite color. In the areas the two logos collide, the original color scheme they share can be seen because the encryption key was the same for both pictures.

(Code - TTP)

```
result = ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(pic1, key))
result2 = ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(pic2, key))
result3 = ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(pic1, pic2))
```

Task V. Modes of Operation

Once again, view the resulting ciphertexts. *What do you observe? Are you able to derive any useful information about from either of the encrypted images? What are the causes for what you observe?*

A: The ECB encrypted image still retains a lot of information about the logo. It may not be the same color as the original image, but the shape and outline of the mustang is still distinguishable. On the other hand, the CBC encrypted image looks very similar to the ones we saw earlier after using OTP. There is just noise that covers up any information about the original image. The initialization vector (IV) and resulting ciphertexts made it so we could replicate the OTP without having a key the size of the file.

(Code - CBC)

```
obj = AES.new(key)
cip = obj.encrypt(bytes([a ^ b for a,b in zip(data, iv)]))
cbc = cip
```

```
for l in range(0, loops-1):
    firstByte = lastByte
    lastByte += BLOCK_SIZE
    data = one[firstByte:lastByte]

    xor = bytes([a ^ b for a,b in zip(data, cip)])
    cip = obj.encrypt(xor)
    cbc += cip
```

Task VI. The Limits of Confidentiality

Why was this attack possible? What would this scheme need in order to prevent such an attack?

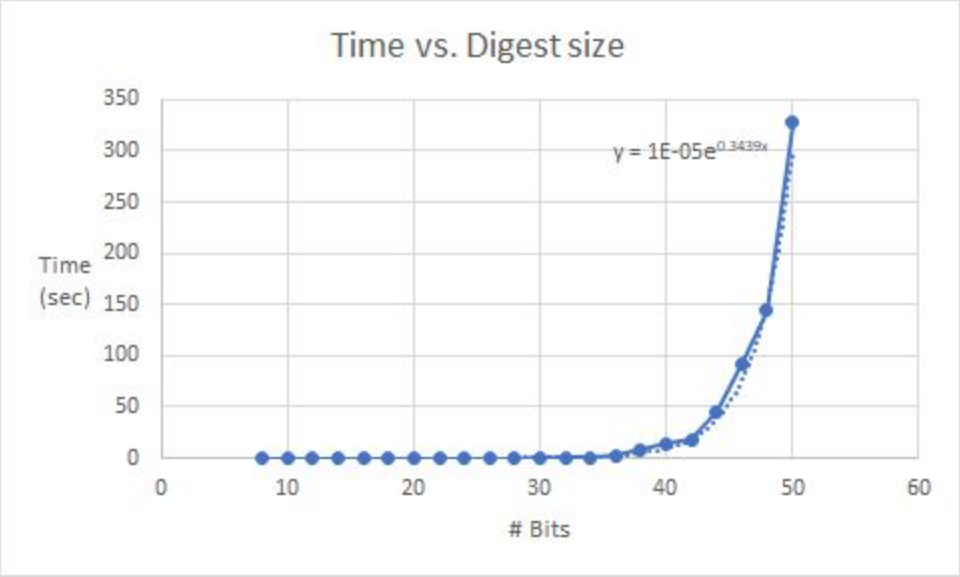
A: The attack was possible because we had access to the original message and the ciphertext. By XOR a part of the original message and the ciphertext, we can reset the bits that are there to zeroes. By doing so, we can XOR the result to the message we want to place there. When the ciphertext is decoded, the original message will display the change we made clear as day. To prevent an attack, the decryption should know the exact phrases to look for in the resulting web cookie, and a tag that would verify its' authenticity.

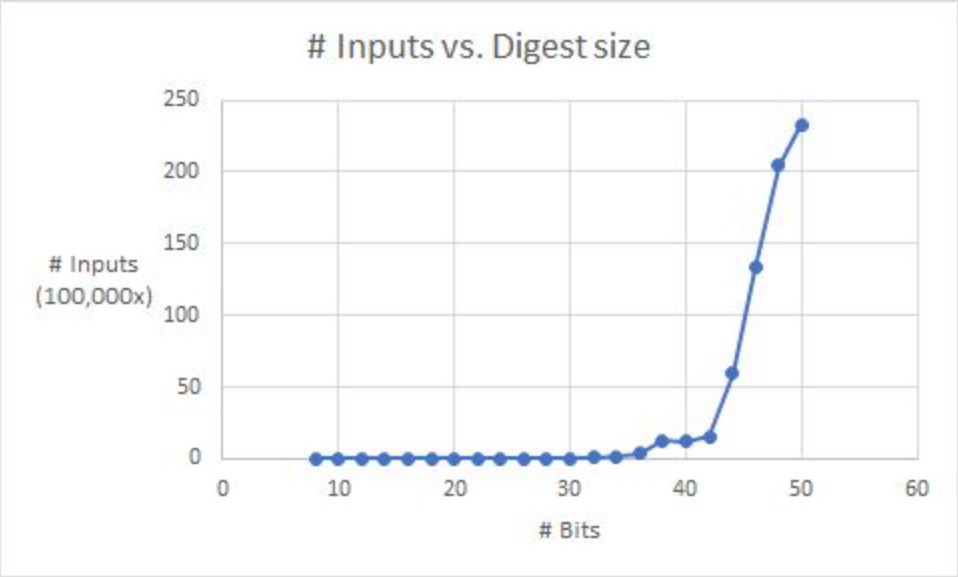
Task VII. Exploring Pseudo-Randomness and Collision Resistance

What do you observe? How many of the bytes are different between the two digests?

A: Almost all the bytes are different between the two digests. It's as if there was nothing similar between the two sentences.

Measure both the *number of inputs* and the *total time* for a collision to be found. Create two graphs: one which plots digest size (along the x-axis) to collision time (y-axis), and one which plots digest size to number of inputs. *Include these graphs in your report.*





What is the maximum number of files you would ever need to hash to find a collision on an n-bit digest?
A: $2^n + 1$ files

Given the Birthday Bound, what is the expected number of hashes before a collision on an n-bit digest?
A: $2^{n/2}$ files

Is this what you observed? Given the data you've collected, speculate on how long it might take to find a collision on the full 256-bit digest.
A: On average, my data showed that most of my collisions happened either at that mark or just below. My exponential trendline for time vs. digest size gave me an equation of:
$y = 1E\text{-}05 * e^{0.3439x}$. If I plug in 256 for x, I get an estimate time of 1.72E33 seconds.

Given an 8-bit digest, would you be able to break the one-way property (i.e. can you find any *pre-image)? Do you think this would be easier or harder (i.e. more or less work) than finding a collision? Why or why not?*
A: Yes, an 8-bit digest should be broken by one-way property. It may be easier at smaller bit numbers because the amount of possibilities is small enough that finding a match of a constant digest becomes more plausible (assuming true randomness).