Student: Ya Zou
Date: Oct.26.2023

Assignment 1 report

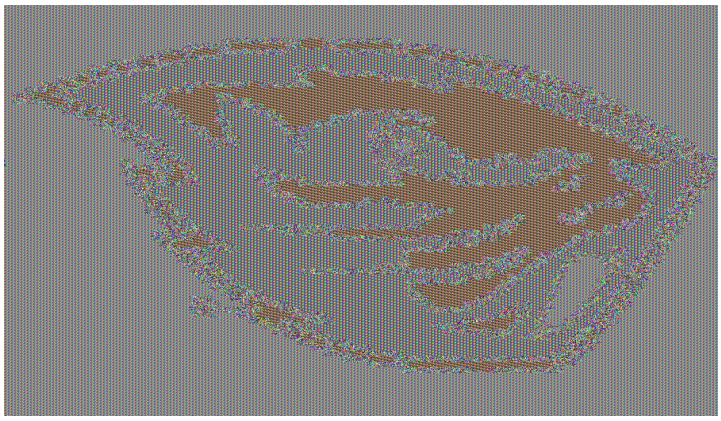
3.1 Encryption Mode - ECB vs. CBC

3.1.2. Display the encrypted picture using any picture viewing software. Can you derive any useful information about the original picture from the encrypted picture? Why or why not? Record both encrypted pictures and the original pictures for your report.

Picture 1: The original picture of the beaver head:

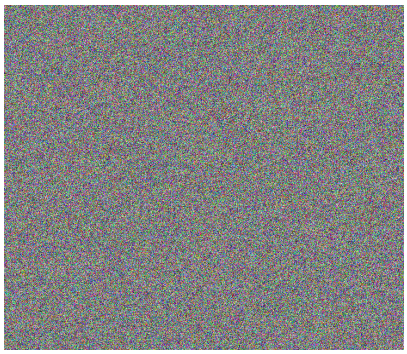
Beaver head CBC mode:

Beaver Head ECB mode

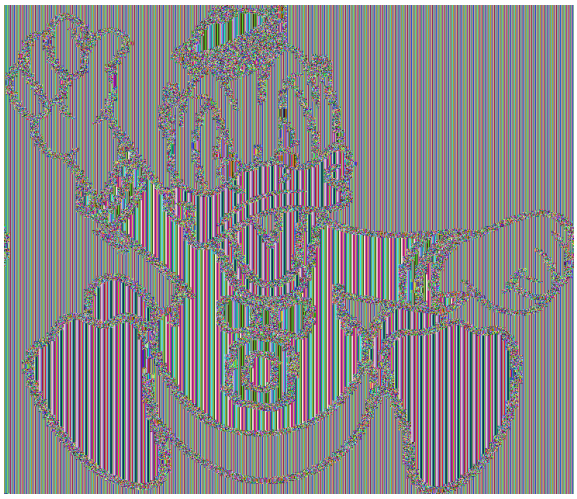


Picture 2: The original picture I chose myself:

Duck CBC Encryption Mode:



Duck ECB Encryption picture:



Based on the pictures above, we can tell that I encrypted in two different ways: CBC and ECB modes. When it's encrypted with CBC mode, the pictures look very random and not easy to figure out. You can't see much from them. But when they're encrypted with ECB mode, you can actually see some patterns from the original pictures.
The reason for these differences is how they're encrypted. In ECB mode, the picture gets divided into small chunks, and each chunk gets encrypted separately. So, you can spot patterns and things from the original picture in each chunk. In CBC mode, though, one chunk's encryption depends on the one before it. This makes it harder to see any patterns, and even small changes in the original image end up looking very different in the encrypted one. This makes it more secure.

3.3 Weak versus Strong Collision Resistance Property

3.32 For the problem above how many trials on average should take one to break the weak collision resistance property using the brute-force method? How many trials did it take for your code to do it? You should repeat your experiment multiple times and report your average trial.

For the problem mentioned, on average, it took about **2365756.60** iterations to breach the weak collision resistance property using the brute-force method. My code required **15** trials for this accomplishment.

```
def weak_collision():

    digest = hashes.Hash(hashes.SHA256())
    digest.update(b"abc")
    weak_hash_value = digest.finalize()[:3]

    for i in range(15):
        time = 0
        total_iterations = 0
        while True:
            # create 3 bytes random chars
            random_characters = os.urandom(3)
            # print(random_characters)
            digest = hashes.Hash(hashes.SHA256())
            #The bytes to be hashed.
            digest.update(random_characters)
            #Finalize the current context and return the message digest as
            bytes.
            weak_hash_value_2 = digest.finalize()[:3]
            time += 1
```

3.33 For the problem above how many trials on average should it take one to break the strong collison resistance property using the brute force method? How many trials did take for your code to do it? You should repeat your experiment multiple times(15 or more) and report your average number of trials

In the context mentioned earlier, there is an average of 4648.875 iterations required to break the strong collision resistance property using the brute force method. My code successfully achieved this in 16 trials.

3.3.4 Based on your observation, which property is easier to break using the brute-force method?

According to my observation, strong collision resistance property is easier to break using the brute-force method

3.3.5 Can you explain your observations above?

yes, the idea of strong collision resistance property is all about uncovering two messages that yield the same hash value, which we call a "strong collision." Think of this like the "birthday paradox." This paradox tells us that in a group of items, such as hash values, the chances of finding two items that match (a strong collision) increase more quickly than you might expect without doing the math.

Now, when it comes to finding a message that shares the same hash value as one you already know (that's the "weak collision"), it's a bit more like a targeted search. But here's the thing, it's less likely to succeed compared to finding a strong collision.

To sum it up, for an attacker, it's simpler to discover a pair of messages with the same hash value (strong collision) than to pinpoint a message that matches the hash value of a given message (that's called weak collision resistance). And it's even easier than trying to work backward from a hash value to find the original message, which is known as the "one-way property."