Cryptography CW1

November 18 2020

1 Cryptographic Software

1.1

Command: `openssl enc -nosalt -aes-128-cbc -pass pass:cryptorulez2020 -a -in msg.txt`
Result: `lt9NU6xekInB6e4WpqEOOQ==`

1.2

Command: `openssl sha256 enc.pw`
Result: `SHA256(enc.pw)= 9b6427b0ecddf429922fd6e975c76c339b02e52c392772ae83cb40771f769cb5`

Command: `openssl sha256 mac.pw`
Result: `SHA256(mac.pw)= ad12d06829e909e10404c7beca0b93504b4f987294f5300f6d68bb78af8407b8`

1.3

Command: `openssl chacha20 -d -K 9b6427b0ecddf429922fd6e975c76c339b02e52c392772ae83cb4771f769cb05 -iv c0dec0dec0dec0dec0dec0dec0dec0de -a -in encrypted.txt`
Result: Do. Or do not. There is no try. -Yoda

1.4

Command: `openssl sha256 -hmac ad12d06829e909e10404c7beca0b93504b4f987294f5300f6d68bb78af8407b8 78c73589f9 encrypted.txt`
Result: `HMAC-SHA256(encrypted.txt)= 78adcd33398b4914303955697515ba5733abd2926b473a28a2c99cf6211f54b5`

Yes this is correct as it matches the authentication tag

2 Block Ciphers

2.1

2.1.1

$c \to 0$
$9 \to d$
$9 \to d$
$4 \to b$
$f \to a$
$8 \to 1$
$8 \to 1$

Result: 0ddba11

2.1.2

$f \to a \oplus 4 = e$
$4 \to b \oplus 4 = f$
$4 \to b \oplus 1 = a$
$1 \to 7 \oplus b = c$
$b = \text{IV}$

Result: cafe

2.1.3

$e = \text{IV}$
$e \to 5 \oplus 4 = f$
$5 \to 0 \oplus 4 = e$
$0 \to c \oplus 1 = e$
$c \to b \oplus b = d$

Result: feed

2.2

Using OFB Encryption rules we know that:

$a \to ?_1 \oplus X_1 = 5$
$?_1 \to ?_2 \oplus X_2 = 9$
$?_2 \to ?_3 \oplus X_3 = d$
$?_3 \to ?_4 \oplus X_4 = e$
$?_4 \to 1 \oplus e = f$
$1 \to F \oplus 1 = e$
$F \to ?_5 \oplus (X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus e \oplus 1) = 2$

We are told to change the verify function from $e1$ to 10, therefore

$$?_4 \to 1 \oplus 1 = 0$$

$$1 \to f \oplus 0 = f$$

To get the last value, as we know it is a parity digit we can use some XOR properties to reduce the equations

$$?_5 \oplus (X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus e \oplus 1) = 2$$
$$?_5 \oplus (X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus 1 \oplus 0) = ?_6$$
$$e \oplus 2 = ?_6 \oplus 0$$
$$?_6 = c$$

The new code is $a59de0fc$

2.3

We are told that there are 32768 bytes in a block. This block is divided into sub-blocks that are 4 bits long, creating 65,536 sub-blocks. As the cipher only shuffles the blocks around according to the key, we need to unique identify your sub-blocks in order to find what offset they are shuffled to in the ciphertext. As each sub-block is 4 bits wide, we have 16 unique identifiers that we can use. However, as we have to have some sort of base number, there are only 15 unique identifiers (we can use the base 0000). This means we need to run $65,536 \div 15 = 4369$ times. As the remainder is one, the last position where the sub-block never changed from our base number is the offset of the last sub-block. On each run, we submit a string 0001||0010||0011||..||1111. This string starts at the sub-block offset of 0 and moves to the next offset of a multiple of 15. As every run contains 15 unique identifiers, whatever offset the unique identifier in the ciphertext is where the cipher shuffled that sub-block.

3 Semantic Security

3.1

$\pi = (Gen, Enc, Dec)$
For Key Generation, we will use

$$Gen(\lambda) = Gen_1(\lambda) \oplus Gen_2(\lambda)$$
$$Enc(m) = Enc_1(Enc_2(m))$$
$$Dec(m) = Dec_2(Dec_1(m))$$

By XORing the two functions, we preserve the key length while not reducing the randomness of the CPA secure key.

With encryption, we can consider two situations. The simplest is if Enc_1 is IND-CPA secure, then any message encrypted will not be distinguished with non-negligible probability. If Enc_2 is IND-CPA secure, this means that the input can not be distinguished with non-negligible probability. If we further encrypt this non-distinguishable text, we will receive a ciphertext that is also indistinguishable from the plain text.

Decryption must follow the same logic as encryption, as decryption is deterministic. The correctness argument is:

$$m = Dec_2(Dec_1(Enc_1(Enc_2(m))))$$

3.2

Theorem: If π_1 or π_2 is CPA-Secure, then π is also CPA-Secure
Proof by Reduction:
1) Assume π is insecure and that some adversary A breaks π
2) PPT Algorithm B executes A as a blackbox subroutine D. The setup is as follows:

Oracle O that A has access to allowing it access Enc
Gen that allows A to create plaintext, ciphertext pairs

For arbitrary message m via oracle O black box uses Enc to return a ciphertext c.
For challenge message m_0, m_1, oracle O chooses one, uses Enc and returns a ciphertext c

3) Show through $A's$ output that π_1 and π_2 are not CPA-Secure

For CPA security - $|Pr[PrivK^{cpa}_{A,\pi}(\lambda) = 1] - 1/2| \leq negl(\lambda)]$ follows by construction of D. As π is insecure, this means that the A is able to predict without negligible probability the ciphertext of the chosen plaintext. As π is built upon the two existing encryption, π_1 or π_2, this means that neither π_1 or π_2 are CPA-Secure.

However, as know that one of π_1 or π_2 are CPA-Secure, there can be no algorithm B that A can use to break π. This means that adversary A can not break π, proving that π is CPA-Secure.

4 Hash Functions

4.1

Yes, f is collision resistant. By contradiction, assume $f'(x, y)$ is not collision resistant. This implies we can find $y_1 \neq y_2$ such that $f'(x, y_1) = f'(x, y_2)$.

$$f'(x, y_1) = f'(x, y_2)$$
$$E(x, x \oplus y_1) \oplus x = E(x, x \oplus y_2) \oplus x$$
$$E(x, x \oplus y_1) = E(x, x \oplus y_2)$$

Which implies that $y_1 = y_2$ giving us a contradiction. Therefore f has to be collision resistant

4.2

4.2.1

No, H is not collision resistant. Assume F(X) is not collision resistant, this implies we can find $x \neq y$ such that $F(x) = F(y)$.

$$H(x) = F(G(x))||G(F(x))$$
$$H(y) = F(G(y))||G(F(y))$$

Set $H(x) = H(y)$

$$F(G(x))||G(F(x)) = F(G(y))||G(F(y))$$

This implies that

$$F(G(x)) = F(G(y))$$
$$G(F(x)) = G(F(y))$$
$$F(x) = F(y)$$

If $F(x)$ is not collision resistant, then given two different inputs it is possible for $F(G(x)) = F(G(y))$ as G is a collision resistant function. Furthermore it is possible for $F(x) = F(y)$ which means $G(F(x)) = G(F(y)$. The same logic can be applied if we assume $G(x)$ is not collision resistant. This implies that $H(x) = H(y)$ demonstrating H is not collision resistant

4.2.2

Yes, H is collision resistant. By contradiction, assume $H'(X)$ is not collision resistant. This implies we can find $x \neq y$ such that $H'(x) = H'(y)$.

$$H'(x) = F(G(x)||x)||G(F(x)||x)$$
$$H'(y) = F(G(y)||y)||G(F(y)||y)$$

Which implies that

$$F(G(x)||x)||G(F(x)||x) = F(G(y)||y)||G(F(y)||y)$$
$$F(G(x)||x) = F(G(y)||y)$$
$$G(x)||x = G(y)||y$$
$$G(x) = G(y)$$
$$G(F(x)||x = G(F(y)||y)$$
$$F(x)||x = F(y)||y$$
$$F(x) = F(y)$$

As we know that one of the functions F or G is collision resistant, this gives us a contradiction as both functions are equivalent and $x \neq y$. Therefore H has to be collision resistant

4.3

No, H is not collision resistant. Consider an input to the function, x

$$H(x) = h_n$$
$$x = pad(x) = x_1, ..., x_n$$
$$h_n = G(x_i \oplus h_{i-1})$$

Let another input $y = x_n$

$$y = pad(y) = x_n$$
$$G(x_{n-1} \oplus h_{i-1}) = G(y \oplus h_0)$$

As $h_0 = 0^b$

$$x_{n-1} \oplus h_{i-1} = y \oplus 0$$
$$x_n = y$$
$$x_n = x_n$$

There is a collision, therefore H is not collision resistant.