IDATT2503 - Cryptography Assignment 02

Note to TA and teacher: There were some updates made to the assignment regarding keys that were not in their full length in the initial text. I did the assignment prior to the update with new K_2, and have now tried adjusting my answers to be correct with the new key, but there may be some error due to trying to patch up the errors carried over from using the provided wrong key.

Task 1 - AES: A little history lesson

1.1

> The evolutionary history of AES differs from that of DES. Describe how the processes of establishing the two standards differ.

The main difference in how DES (Data Encryption Standard) and AES (Advanced Encryption Standard) was established as encryption algorithm of choice for the US government was mainly in how the process around selecting it was handled.

DES was adoped in 1977, when the then National Bureau of Standards (NBS), now National Institute of Standards and Technology (NIST), was looking for a standard for encryption. The algorithm they landed on was based on the IBM Lucifer algorithm, which was then modified according to input from the National Security Agency (NSA), resulting in DES. The process for coming up with and selecting the DES algorithm was somewhat closed, with little public scrutiny during the development phase. In addition, concerns were raised from encryption and data security communities regarding parts of NSA's involvement which resulted in reducing the key length from 128 to 65 bits, which made it more vulnerable to brute force attacks as the computing power grew. Basically, it was not very future proof.

In comparison, the process for establishing AES as the new standard was highly transparant and open. In 1997, NIST initiated a public call for proposals for a new encryption standard to replace the then aging and vulnerable DES algorithm. Instead of doing it in a closes and almost "in-house" way, as with DES, the new standard was set up more like a competition with international participation. Cryptographers from all over the world submitted their propsals for new algorithms, and the process of evaluating the submissions was public, with multiple rounds of analysis, peer reviews, and public discussions. After a period of years reviewing and analysing different candidates, the algorithm choses was the **Rijndael** cipher. This was developed by the two Belgian cryptographers Joan Daemen and Vincent Rijmen. The AES standard went in effect as the US federal government standard 26. May 2002.

1.2

> What is the name of the algorithm that is known as AES and who developed this algorithm?
>
> Why was it by many a surprise that it was this that was chosen to be the new standard?

As mentioned above, the algorithm known as AES, is originally named **Rijndael**. It was developed by the Belgian cryptographers Joan Daemen and Vincent Rijmen.

The selection of Rijndael as the new standard was a surprise to many, primarliy because it was less well-known and not as widely as other entries, like Twofish and Serpent. These other algorithms were considered to be stronger contenders because of their high levels of security, with more support from, and use by, influental sectors of the cryptographic community and the computer industry. Twofish and Serpent was devloped by cryptographers with established reputations in the community, and were considered very robust and secure ciphers. It's worth taking a closer look at a basic overview of Rinjdael, Twofish and Serpent to understand why Rijndael was the winner.

Serpent was well known for prioritizing security above performance, using a very conservative design with an emphasis on resistance to cryptoanalytic attacks, using 32 rounds of encryption. Compared to Rijndael's 10, 12, or 14 rounds, this made it much more secure, but also much slower.

Twofish was developed by the creators of the widely used Blowfish algorithm, which was then already a popular choice in security applications. The strong repotation of its creator made Twofish a favoured contender. Twofish employed complex key scheduling and block structures, making it very secure, but its performance in hardware implementation was much lower than that of the Rijndael cipher.

In the end, both Serpent and Twifish were viewed as higly secure, and their lower performance was often regarded as a reasonable trade-off for better security. Combined with being developed by well-known people, this made them the obvious choices in the eyes of many.

Rijndael was choses due to its balanced combination of security, efficiency, and flexibility. In terms of performance, Rijndael had faster encryption and decryption speeds, especially in hardware implementations. Being faster, naturally made it more efficient, especially in environment with constrained or limited resouces, siuch as embedded systems or smart cards, where both processing power and memory is limited. The Rijndael cipher was also developed with both hardware **and** software efficency in mind, which gave it an edge over the competitors that were often optimized for one or the other.

In short, Rijndael managed to find a middle ground of performance, security and flexibility that the others did not. Twofish and Serpent went for a more conservative security model, heavily focusing on cryptographich strength. While it could be said to be a "weaker" encryption than the two favoured candidates, Rijndael managed to provide strong security without sacrificing performance, making it more suitable for an all-round algorithm. This enabled implementation accross a broad range of applications, from high-performance servers to resource-contrained devices and environments.

Task 2 - Diffusion and confusion

Bits to use for messages and keys: `128`

AES: Standard settings, 128-bits key, and no chaining.

Diffusion: The principle of spreading out the influence of a single plaintext bit across many ciphertext bits. This means that a small change in the plaintext should lead to a significantly different ciphertext.

Keys in hexadecimal

$K_1 = 0123456789ABCDEF0123456789ABCDEF$

$K_2 = 1123456789ABCDEF0123456789ABCDEF$

Plaintext in hexadecimal

$x_1 = 01000000000000000000000000000000$
$x_2 = 02000000000000000000000000000000$

Ciphers:

```
A One-time pad (XOR) (see lecture notes)
B Affine cipher, use same key for both compoenents (see lecture notes)
C One round of AES
D Full AES.
```

2.a

> For each cipher above, encrypt both x_1 and x_2, using the key K_1 and compare the results, with regards to diffusion.

A - One-time pad (XOR)

One-Time Pad (OTP) is done by doing a XOR operation on the plaintext with the key:

- Encrypting x_1: `plaintext1 XOR K1`
- Encrypting x_2: `plaintext2 XOR K1`

Since x_1 and x_2 differs only in the first byte (`01` vs `02`), the XOR operation will change only this first byte between the two plaintexts. XOR compares the bits of two binary numbers and produces a result based on fixed and known rules:

- If the two bits are the same (both 0 or both 1), the result is 0.
- If the two bits are different (one 0 and the other 1), the result is 1.

In this application, it will produce a change in only the first byte. Running the OTP and XOR operation on the given key1 and plaintexts, the following ciphertexts are produced:

```
x1 XOR K1 = 0023456789abcdef0123456789abcd01
x2 XOR K1 = 0323456789abcdef0123456789abcd01
```

Again, the difference is only in the first byte, `00` in x_1 vs. `03` in x_2.

Diffusion: Minimal, for the above stated reasons of only changing the first byte in the two very similar plaintexts.

B - Affine cipher, same key for both components

The Affine cipher works by applying a linear transformation to each block of data, and can be defined as

$$C = (a \times P + b) \bmod m$$

Where:

- C is the ciphertext.
- P is the plaintext.
- a and b are constants derived from the key.
- $m = 256$ since we are working with 8-bit bytes, giving 256 possible values.

Diffusion:

For simplicity's sake, using an online Affine encryptor, it's clear that the diffusion is minimal, similar to the XOR operation.

```
Affine x1 = 24232323232323232323232323232323
Affine x2 = 25232323232323232323232323232323
```

The only difference is from where the input text was different. Everything else stays identical.

Using the same key for both components means applying the same linear transformation to each block. Since `x1` and `x2` differ in only one bit, this difference will be present in the output, but the diffusion is minimal, as the affine transformation doesn't have the complexity of more advanced ciphers like AES where you would change the entire message based on more complex rules.

Meta comment: All that said, it was my understanding from theprevious assignment that the Affine cipher worked with letters in the alphabet. So I'm a little confused. But from what I gather, the "solution" here is to split the hexadecimal string into smaller components and convert them to an integer and assign these values to a and b. The above solution and explanation is a combination of results by online sources (GPT, Wikipedia, and various cipher calculators (for lack of a better term)).

C - One round of AES:

Assuming "One round AES" is `Number of Rounds: 1` on https://legacy.cryptool.org/en/cto/aes-step-by-step

x_1 encoded with K_1:

```
01fcf41f 4c13eaaa 96747c97 c49b6222
```

x_2 encoded with K_1

```
19fcf41f 4c13eaaa 96747c97 c49b6222
```

Diffusion: Minimal. The only change from x_1 to x_2 is that the first two characters changed from `01` to `19`. The rest of the ciphertext remains identical accross x_1 and x_2

D - Full AES

Assuming "Full AES" is `Number of Rounds: 10` , i.e. defualt setting at https://legacy.cryptool.org/en/cto/aes-step-by-step

x_1 encoded with K_1:

```
0694267b a398480c 6b2b9f64 9be476cb
```

x_2 encoded with K_1

```
282f7b11 019800f8 a978c6f7 50827ab5
```

Diffusion: A single bit change in plaintext will result in a completely different ciphertext.

Summary

- **XOR**: Minimal diffusion. Only the affected byte in the plaintext influences the corresponding byte in the ciphertext.
- **Affine Cipher**: Minimal diffusion. The change is localized in the first byte only, and the rest of the ciphertext remains identical.
- **One round of AES**: Moderate diffusion. The first few bytes of the ciphertext change, but still very similar to the other ciphertext.
- **Full AES**: Complete diffusion. A small change in the plaintext results in a completely different ciphertext, with changes spread across all bytes.

2.b

> For each cipher, encrypt x1 using K1 and K2 , and compare the results. How many bits change?

Plaintext:

- $x_1 = 01000000000000000000000000000000$

Keys:

- $K_1 = 0123456789ABCDEF0123456789ABCDEF$
- $K_2 = 1123456789ABCDEF0123456789ABCDEF$

A - One-time pad (XOR): 51 bits change.

- Ciphertext for x1 using K1: `0023456789abcdef0123456789abcd01`
- Ciphertext for x1 using K2: `0323456789abcdef0123456789abcd01`
- Number of differing bits: 2 bits change.

B - Affine cipher: 1 bit changes.

- Ciphertext for x1 using K1: `24232323232323232323232323232323`
- Ciphertext for x1 using K2: `34232323232323232323232323232323`
- Number of differing bits: 1 bit changes.

C - One round of AES: 2 bits change.

- Ciphertext for x1 using K1: `01fcf41f4c13eaaa96747c97c49b6222`
- Ciphertext for x1 using K2: `b8fcf41f5c13eaaa86747c97d49b6222`
- Number of differing bits: 8 bits change.

D - Full AES: 61 bits change.

- Ciphertext for x1 using K1: `0694267ba398480c6b2b9f649be476cb`
- Ciphertext for x1 using K2: `282f7b11019800f8a978c6f750827ab5`
- Number of differing bits: 61 bits change.

Task 3 - Cryptohack Challenges

Completed challenges up to "Symmetric Ciphers - How AES Works: 7 - Bringing it all together", submitting the flags in the challenge system.

Introduction to CryptoHack

Challenges and flags:

Challenge	Flag
Finding Flags	`crypto{y0ur_f1rst_fl4g}`
Great Snakes	`crypto{z3n_0f_pyth0n}`
ASCII	`crypto{ASCII_pr1nt4bl3}`
Hex	`crypto{You_will_be_working_with_hex_strings_a_lot}`

Challenge	Flag
Base64	`crypto/Base+64+Encoding+is+Web+Safe/`
Bytes and Big Integers	`crypto{3nc0d1n6_4ll_7h3_w4y_d0wn}`
XOR Starter	`crypto{aloha}`
XOR Properties	`crypto{x0r_i5_ass0c1at1v3}`
Favourite byte	`crypto{0x10_15_my_f4v0ur173_by7e}`
You either know, XOR you don't	`crypto{1f_y0u_Kn0w_En0uGH_y0u_Kn0w_1t_4ll}`

Modular Arithmetic

Challenges and flags:

Challenge	Flag
Greatest Common Divisor	1512
Extended GCD	-8404
Modular Arithmetic 1	4
Modular Arithmetic 2	1
Modular inverting	9
Quadratic Residues	8
Legendre Symbol:	See math block
Modular Square Root	See math block
Chinese Remainder Theorem	872
Adrien's Signs	`crypto{p4tterns_1n_re5idu3s}`
Modular Binomials	See math block

Legendre Symbol:

$$93291799125366706806545638475797430512104976066103610269938025709952247020061090804870186195285998727680200979853848718589126765742550855954805290253592144209552123062161458584575060939481368210688629862036958857604707468372384278049741369153506182660264876115428251983455344219194133033177700490981696141526$$

Modular Square Root:

$$
\begin{aligned}
&236233930768304863832777329858048929893213750552050038833827105205373474\\
&786235177964731417681795335907187156004112528991924714607490715161276264\\
&086819962118655952206833803260099131188222401602122267224313936218046123\\
&264673246584884042545825793088785658337960096776173859678287785131848935\\
&567982281315512304570528511209944814642675511016000251559241885043210364\\
&181581107154845628426350780558944507365756538185052136796967569976075531\\
&078462357707644003774768176030243492493211364006173877760119462224419275\\
&802418085391624442725406544196255728257284916277274079898964794864520734\\
&\qquad\qquad 97374574454404050571568975083685319391 20
\end{aligned}
$$

Modular Binomials:

```
crypto{112274000169258486390262064441991200608556376127408952701514962644340921899196091557519382763356534106376906489445103255177593594898966250176773605432765983897105047795619470659157057093771407309168345670541418772427807148039207489900810013783673957984006269120652134007689272484517805398390277308001719431273,
132760587806365301971479157072031448380135765794466787456948786731168095877956875295282661565488242190731593282663694728914945967253173047324353981530949360031535707374701705328450856944598803228299967009004598984671293494375599408764139743217465012770376728876547958852025425539298410751132782632817947101601}
```

Symmetric Cryptography

Challenge flags

Challenge	Flag
Keyed Permutations	`crypto{bijection}`
Resisting Brutefrce	`crypto{biclique}`
Structure of AES	`crypto{inmatrix}`
Round Keys	`crypto{r0undk3y}`
Confusion through Substitution	`crypto{l1n34rly}`
Diffusion through Permutation	`crypto{d1ffUs3R}`
Bringing It All Together	`crypto{MYAES128}`

Screenshot from Cryptochallenge