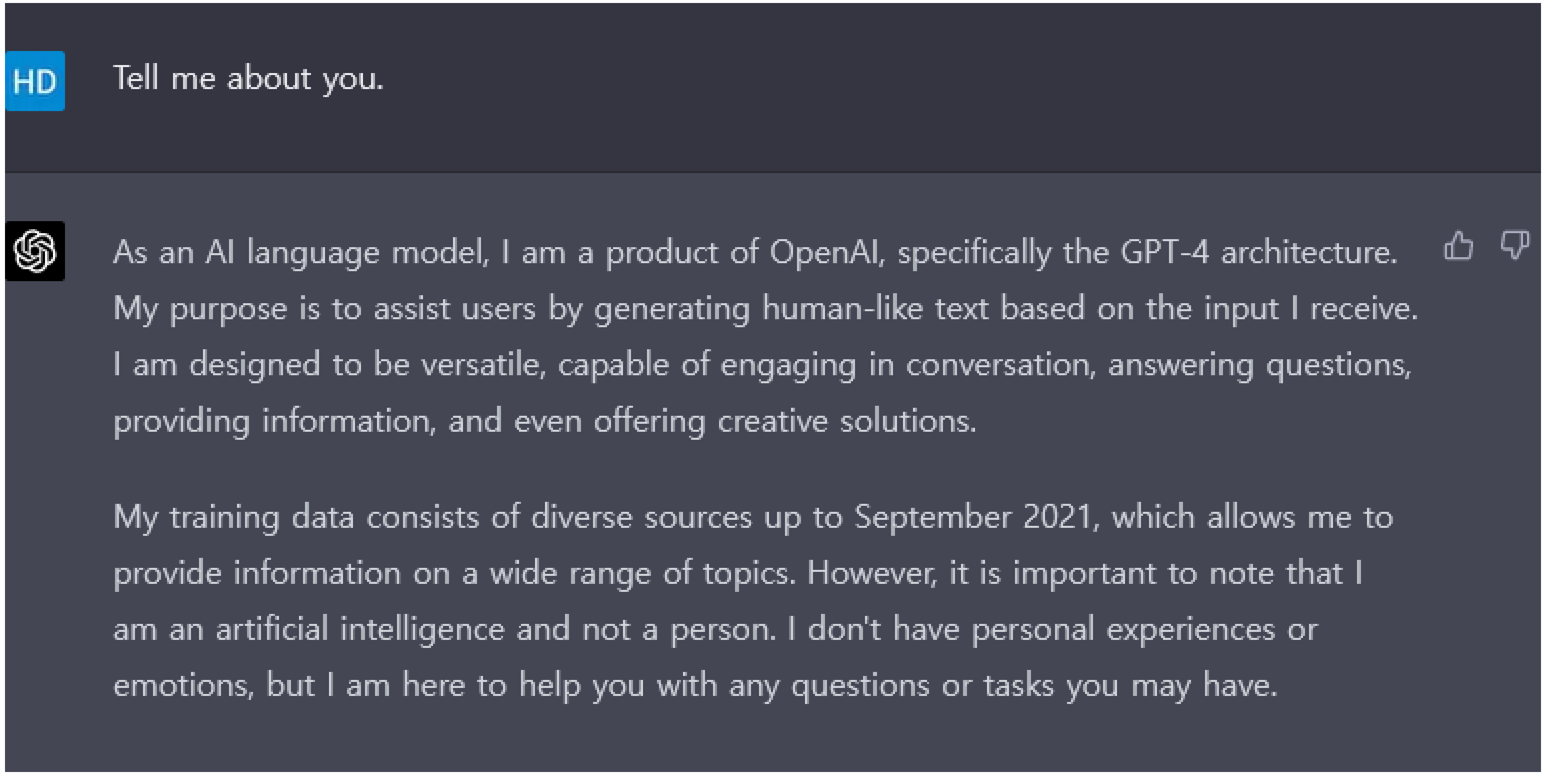
ChatGPT를 사용한 암호 구현

커피동아리 권혁동

ChatGPT란?

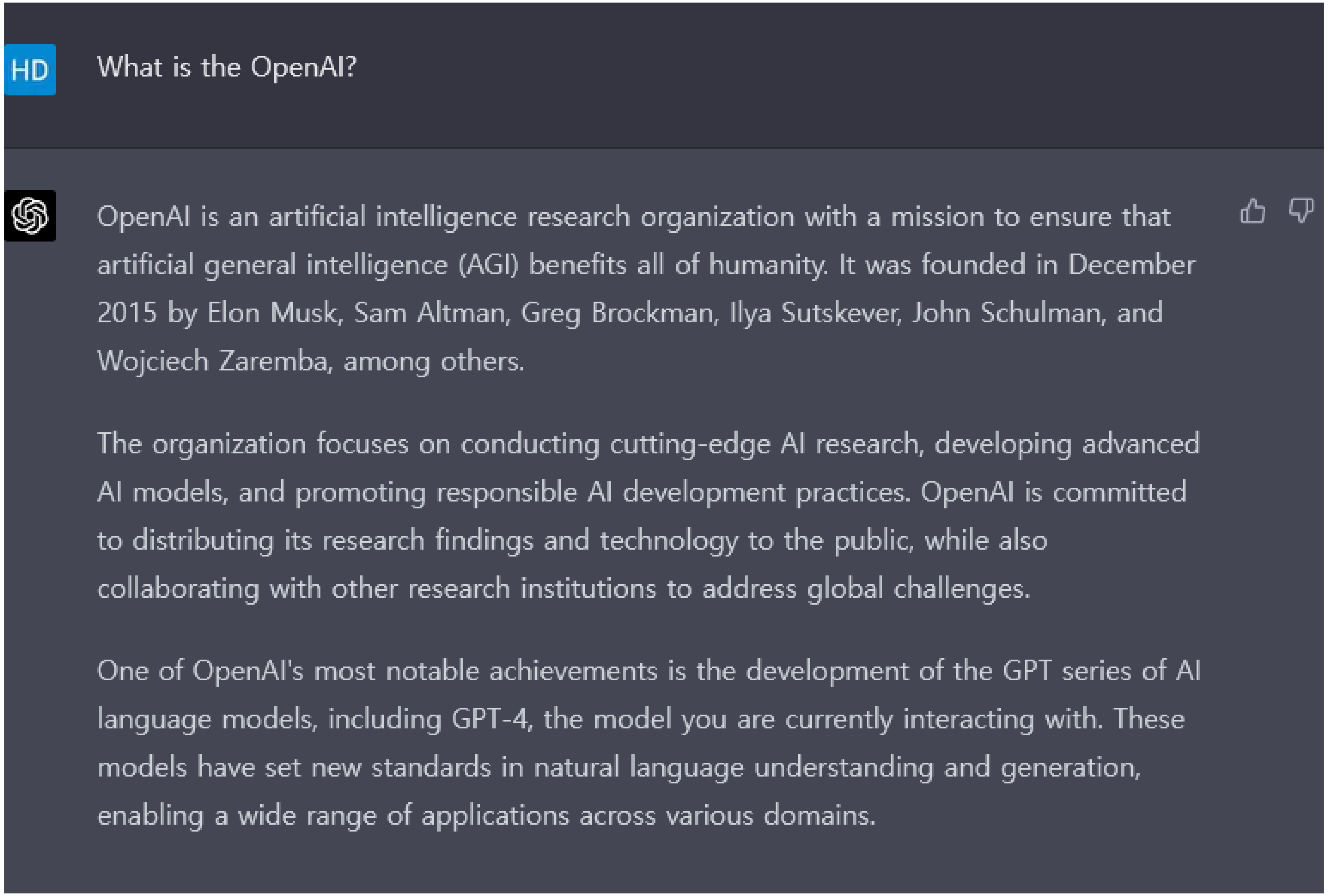


Tell me about you.

As an AI language model, I am a product of OpenAI, specifically the GPT-4 architecture. My purpose is to assist users by generating human-like text based on the input I receive. I am designed to be versatile, capable of engaging in conversation, answering questions, providing information, and even offering creative solutions.

My training data consists of diverse sources up to September 2021, which allows me to provide information on a wide range of topics. However, it is important to note that I am an artificial intelligence and not a person. I don't have personal experiences or emotions, but I am here to help you with any questions or tasks you may have.

ChatGPT란?



What is the OpenAI?

OpenAI is an artificial intelligence research organization with a mission to ensure that artificial general intelligence (AGI) benefits all of humanity. It was founded in December 2015 by Elon Musk, Sam Altman, Greg Brockman, Ilya Sutskever, John Schulman, and Wojciech Zaremba, among others.

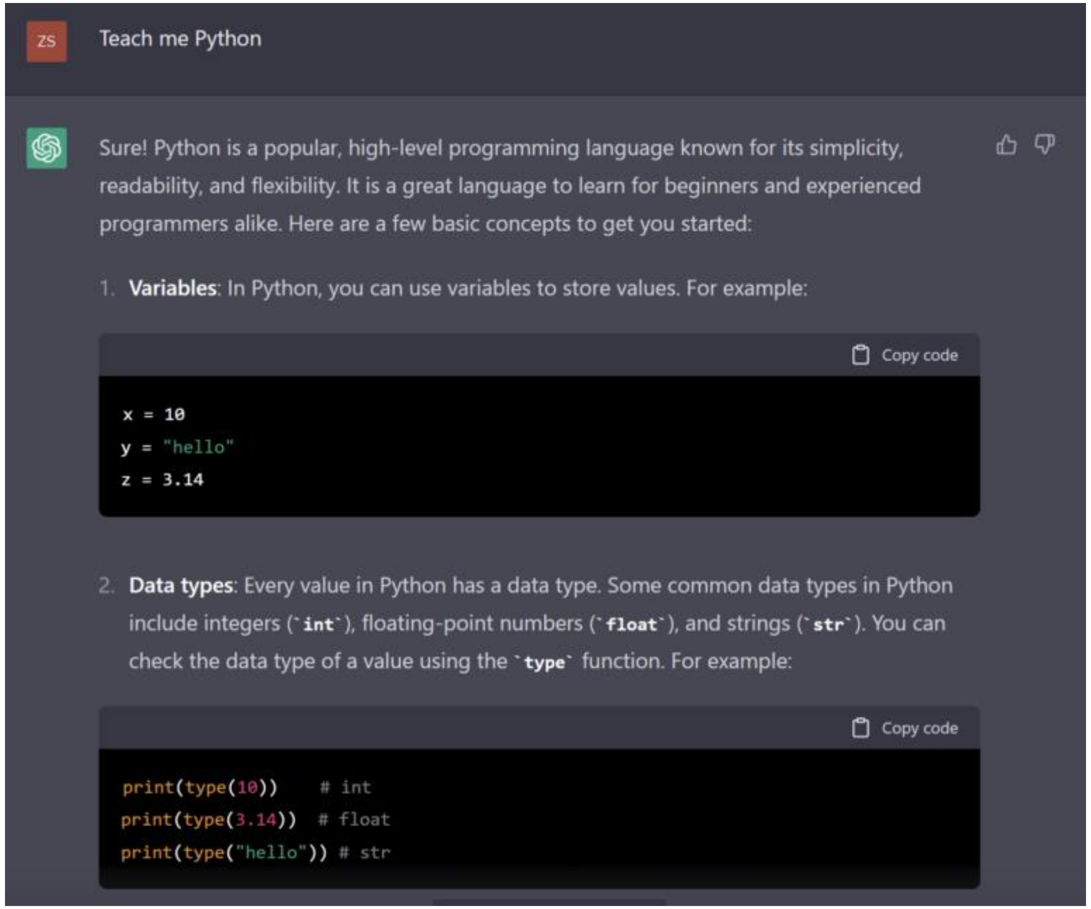
The organization focuses on conducting cutting-edge AI research, developing advanced AI models, and promoting responsible AI development practices. OpenAI is committed to distributing its research findings and technology to the public, while also collaborating with other research institutions to address global challenges.

One of OpenAI's most notable achievements is the development of the GPT series of AI language models, including GPT-4, the model you are currently interacting with. These models have set new standards in natural language understanding and generation, enabling a wide range of applications across various domains.

ChatGPT란?

- OpenAI에서 개발한 대화형 인공지능 서비스
- 무료와 유료 두 가지가 존재
  - GPT-4.0은 유료만 사용 가능, 3시간에 25개의 답변을 받을 수 있음
- 다양한 대화를 할 수 있으며, 창의적인 활동도 가능
  - 디자인
  - 일정 계획
  - 끝말잇기
  - 프로그래밍…

ChatGPT란?



Teach me Python

Sure! Python is a popular, high-level programming language known for its simplicity, readability, and flexibility. It is a great language to learn for beginners and experienced programmers alike. Here are a few basic concepts to get you started:

1. **Variables**: In Python, you can use variables to store values. For example:

```
x = 10
y = "hello"
z = 3.14
```

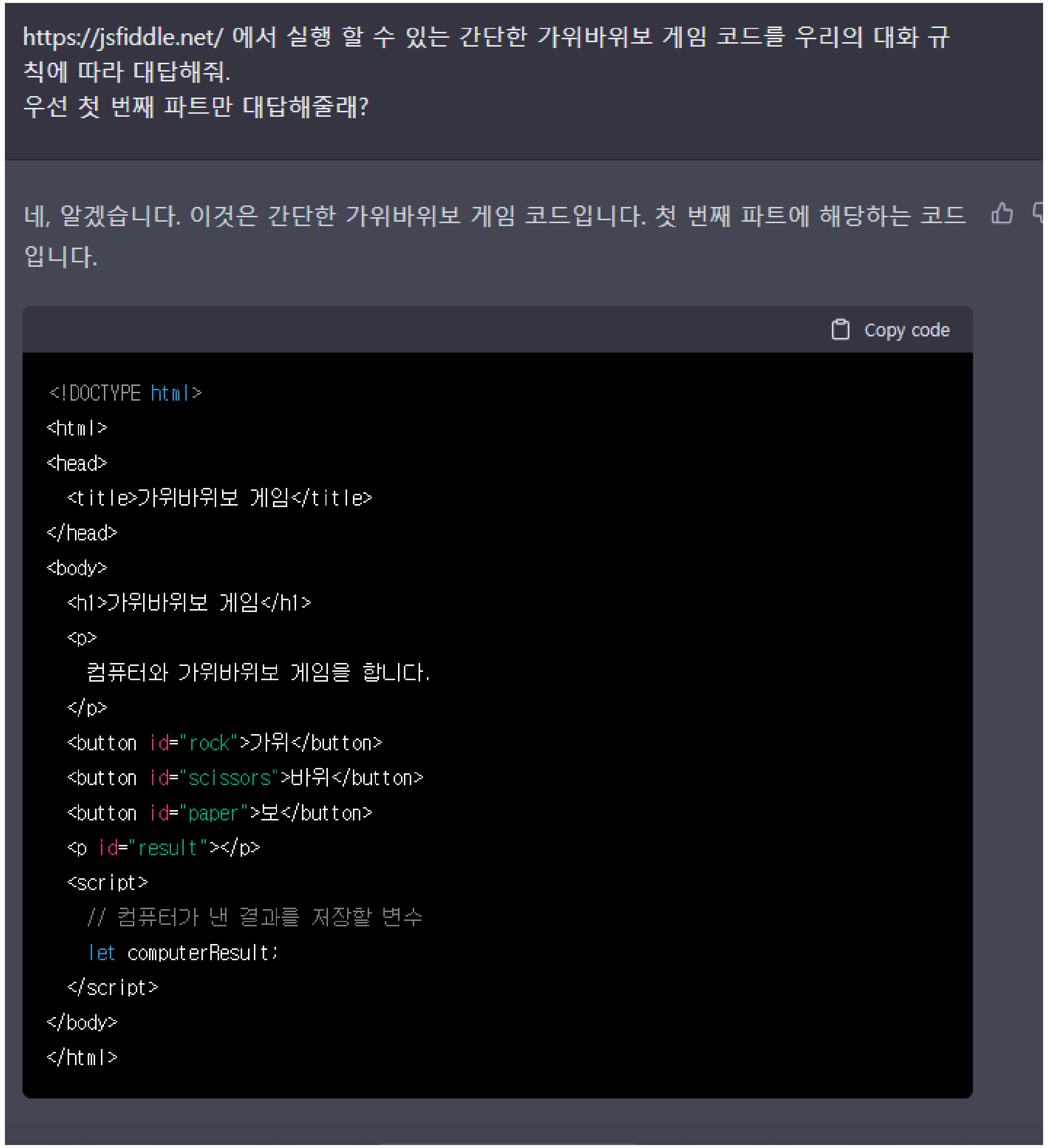
2. **Data types**: Every value in Python has a data type. Some common data types in Python include integers (`int`), floating-point numbers (`float`), and strings (`str`). You can check the data type of a value using the `type` function. For example:

```
print(type(10))     # int
print(type(3.14))   # float
print(type("hello")) # str
```

ChatGPT를 사용한 프로그래밍

- ChatGPT를 사용하여 프로그래밍을 하는 시도가 많아짐
- 장점
  - 프로그래밍을 몰라도 프로그램을 짤 수 있다
  - 프로그래밍을 할 줄 알지만 특정 언어를 몰라도 프로그램을 짤 수 있다
- 단점
  - 프로그래밍을 할 줄 모르면 에러가 있는 코드를 어떻게 고쳐야 할지 모른다
  - 위 단점은 ChatGPT에게 요청하면 에러를 수정해주기도 한다
  - **프로그래밍을 모르면 에러는 없지만, 오동작하는 코드를 찾을 수 없다**

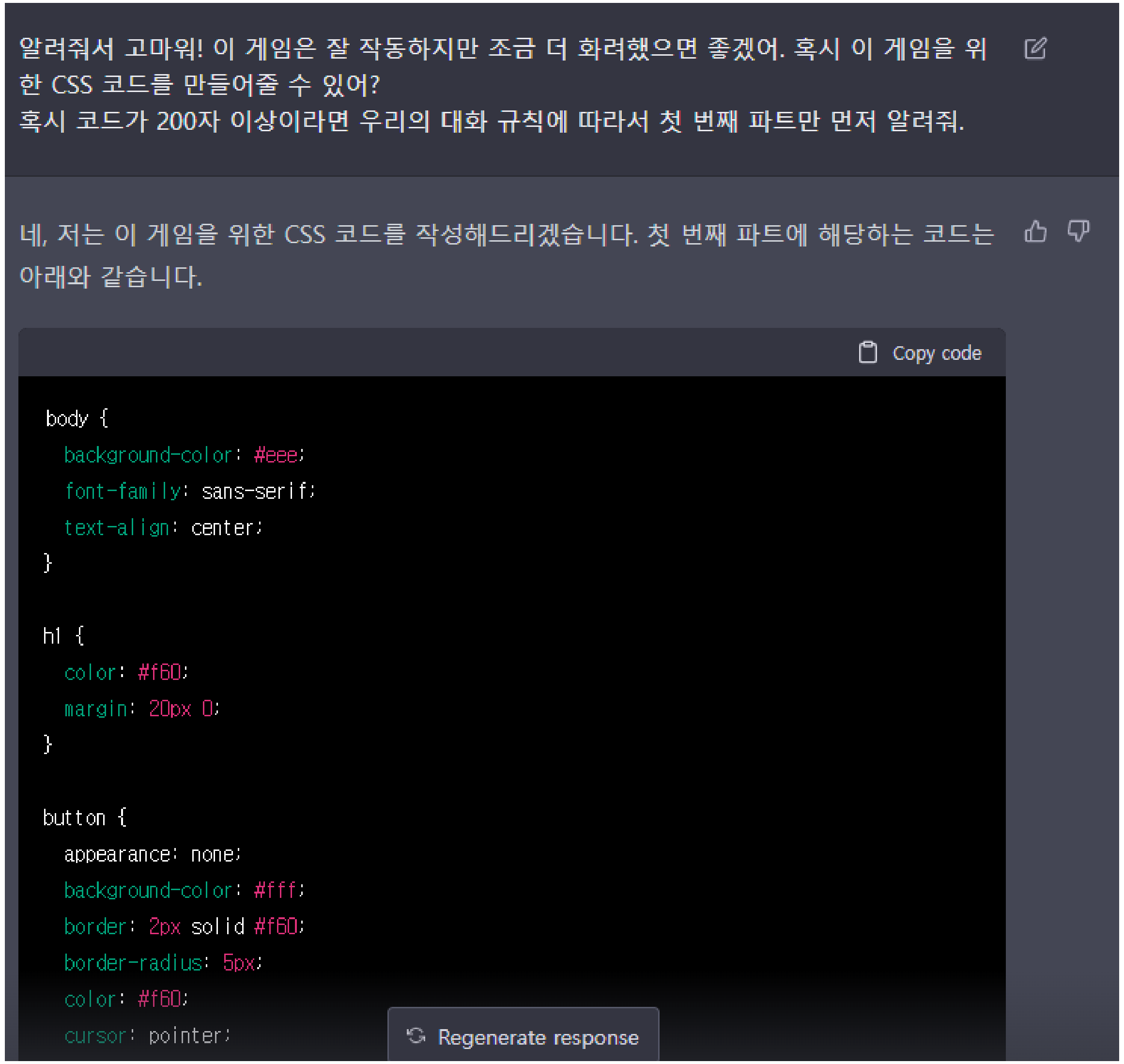
ChatGPT를 사용한 프로그래밍



https://jsfiddle.net/ 에서 실행 할 수 있는 간단한 가위바위보 게임 코드를 우리의 대화 규칙에 따라 대답해줘.
우선 첫 번째 파트만 대답해줄래?

네, 알겠습니다. 이것은 간단한 가위바위보 게임 코드입니다. 첫 번째 파트에 해당하는 코드입니다.

```
<!DOCTYPE html>
<html>
<head>
  <title>가위바위보 게임</title>
</head>
<body>
  <h1>가위바위보 게임</h1>
  <p>
    컴퓨터와 가위바위보 게임을 합니다.
  </p>
  <button id="rock">가위</button>
  <button id="scissors">바위</button>
  <button id="paper">보</button>
  <p id="result"></p>
  <script>
    // 컴퓨터가 낸 결과를 저장할 변수
    let computerResult;
  </script>
</body>
</html>
```



알려줘서 고마워! 이 게임은 잘 작동하지만 조금 더 화려했으면 좋겠어. 혹시 이 게임을 위한 CSS 코드를 만들어줄 수 있어?
혹시 코드가 200자 이상이라면 우리의 대화 규칙에 따라서 첫 번째 파트만 먼저 알려줘.

네, 저는 이 게임을 위한 CSS 코드를 작성해드리겠습니다. 첫 번째 파트에 해당하는 코드는 아래와 같습니다.

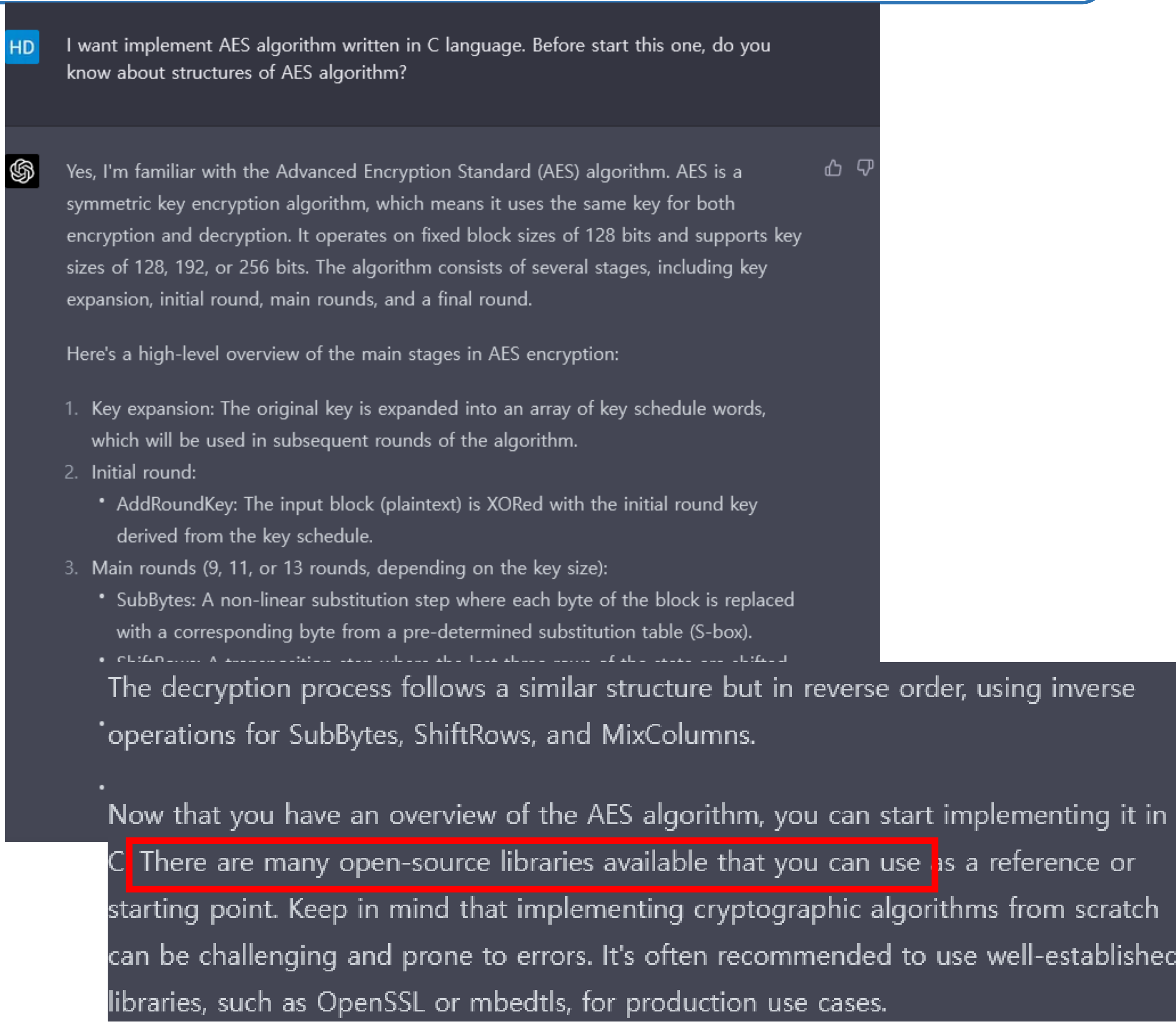
```
body {
  background-color: #eee;
  font-family: sans-serif;
  text-align: center;
}

h1 {
  color: #f60;
  margin: 20px 0;
}

button {
  appearance: none;
  background-color: #fff;
  border: 2px solid #f60;
  border-radius: 5px;
  color: #f60;
  cursor: pointer;
```

ChatGPT와 암호 알고리즘 구현

- 그렇다면 ChatGPT를 사용한 암호 알고리즘은 구현이 가능할까?
  - 기본적으로 ChatGPT는 **언어 모델**
  - 프로그래밍을 위한 AI는 아님
  - 하지만 지금까지 ChatGPT는 프로그래밍과 관련된 많은 부분을 학습함
- 다른 프로그래밍을 보면 암호 알고리즘 구현도 가능할 것으로 보임
  - 구현 대상을 크게 두 종류로 나눔
  - 널리 잘 알려진 알고리즘이며 공개된 소스도 많음: AES
  - 국내에서만 유명한 알고리즘이며 소스코드도 찾기 어려움: CHAM

ChatGPT와 암호 알고리즘 구현: AES

- AES 알고리즘을 알고 있다
  - GPT에게 알고리즘 설명은 생략
- 오픈소스 사용을 추천함
- 직접 구현을 요청



HD I want implement AES algorithm written in C language. Before start this one, do you know about structures of AES algorithm?

Yes, I'm familiar with the Advanced Encryption Standard (AES) algorithm. AES is a symmetric key encryption algorithm, which means it uses the same key for both encryption and decryption. It operates on fixed block sizes of 128 bits and supports key sizes of 128, 192, or 256 bits. The algorithm consists of several stages, including key expansion, initial round, main rounds, and a final round.

Here's a high-level overview of the main stages in AES encryption:

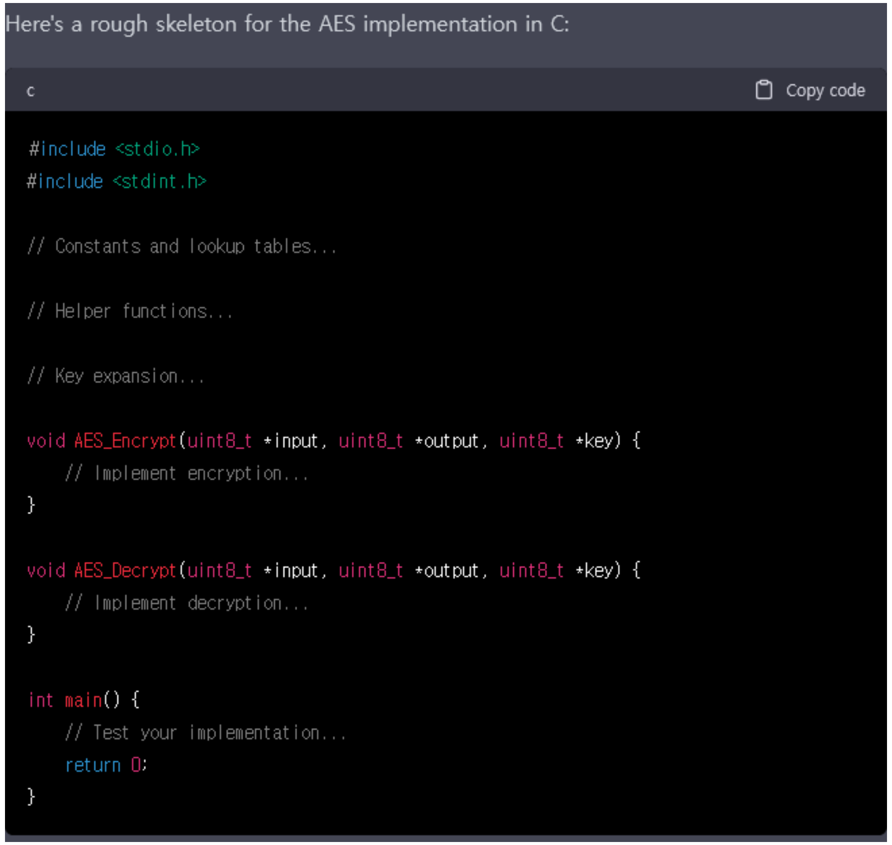
1. Key expansion: The original key is expanded into an array of key schedule words, which will be used in subsequent rounds of the algorithm.
2. Initial round:
   - AddRoundKey: The input block (plaintext) is XORed with the initial round key derived from the key schedule.
3. Main rounds (9, 11, or 13 rounds, depending on the key size):
   - SubBytes: A non-linear substitution step where each byte of the block is replaced with a corresponding byte from a pre-determined substitution table (S-box).

The decryption process follows a similar structure but in reverse order, using inverse operations for SubBytes, ShiftRows, and MixColumns.

Now that you have an overview of the AES algorithm, you can start implementing it in C There are many open-source libraries available that you can use as a reference or starting point. Keep in mind that implementing cryptographic algorithms from scratch can be challenging and prone to errors. It's often recommended to use well-establishe libraries, such as OpenSSL or mbedtls, for production use cases.

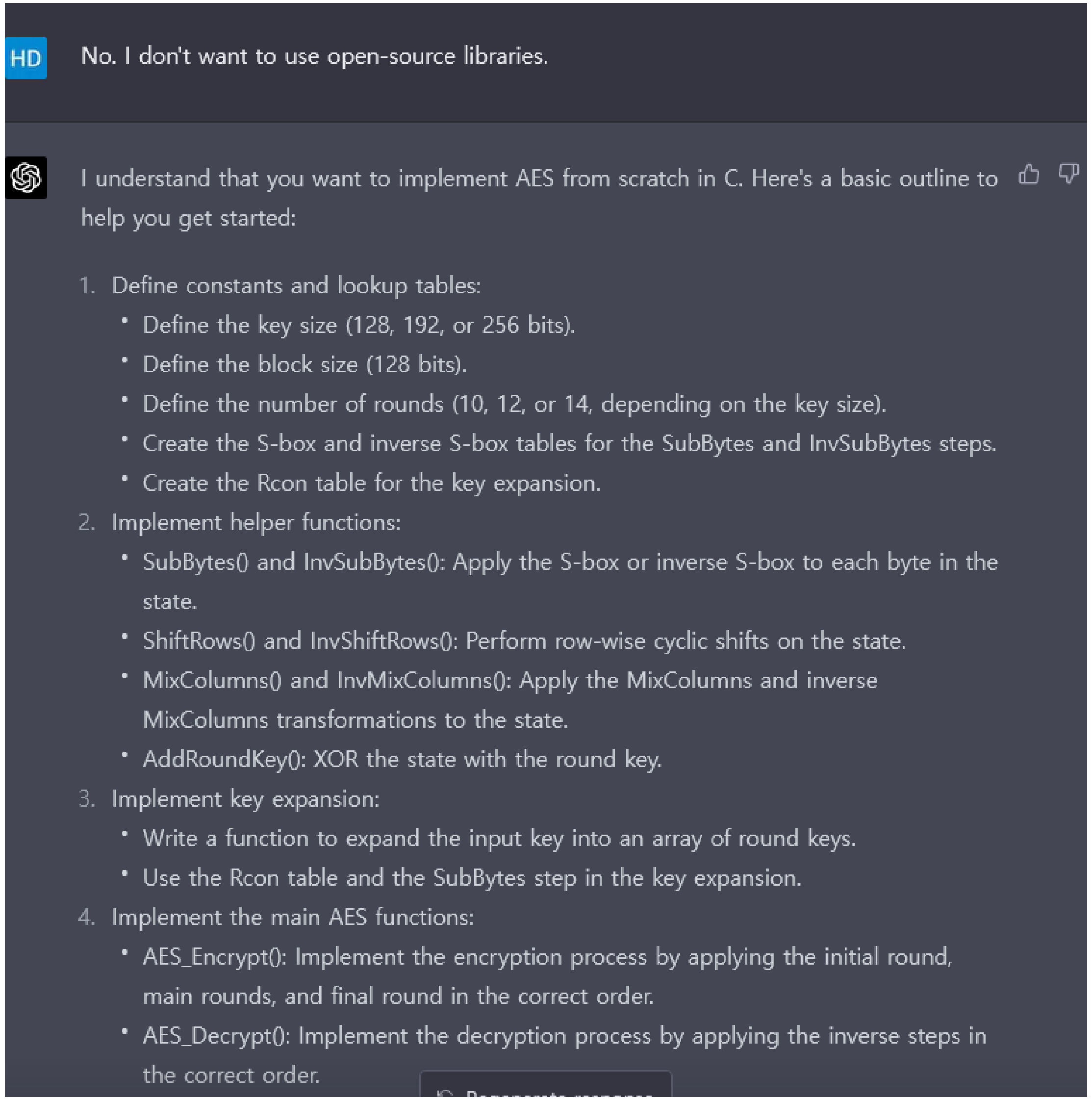
ChatGPT와 암호 알고리즘 구현: AES

- 오픈소스는 제외해달라 요청
    - GPT가 구현 방법을 알려줌
    - 알고리즘 뼈대까지 세워줌



Here's a rough skeleton for the AES implementation in C:

```c
#include <stdio.h>
#include <stdint.h>

// Constants and lookup tables...

// Helper functions...

// Key expansion...

void AES_Encrypt(uint8_t *input, uint8_t *output, uint8_t *key) {
    // Implement encryption...
}

void AES_Decrypt(uint8_t *input, uint8_t *output, uint8_t *key) {
    // Implement decryption...
}

int main() {
    // Test your implementation...
    return 0;
}
```



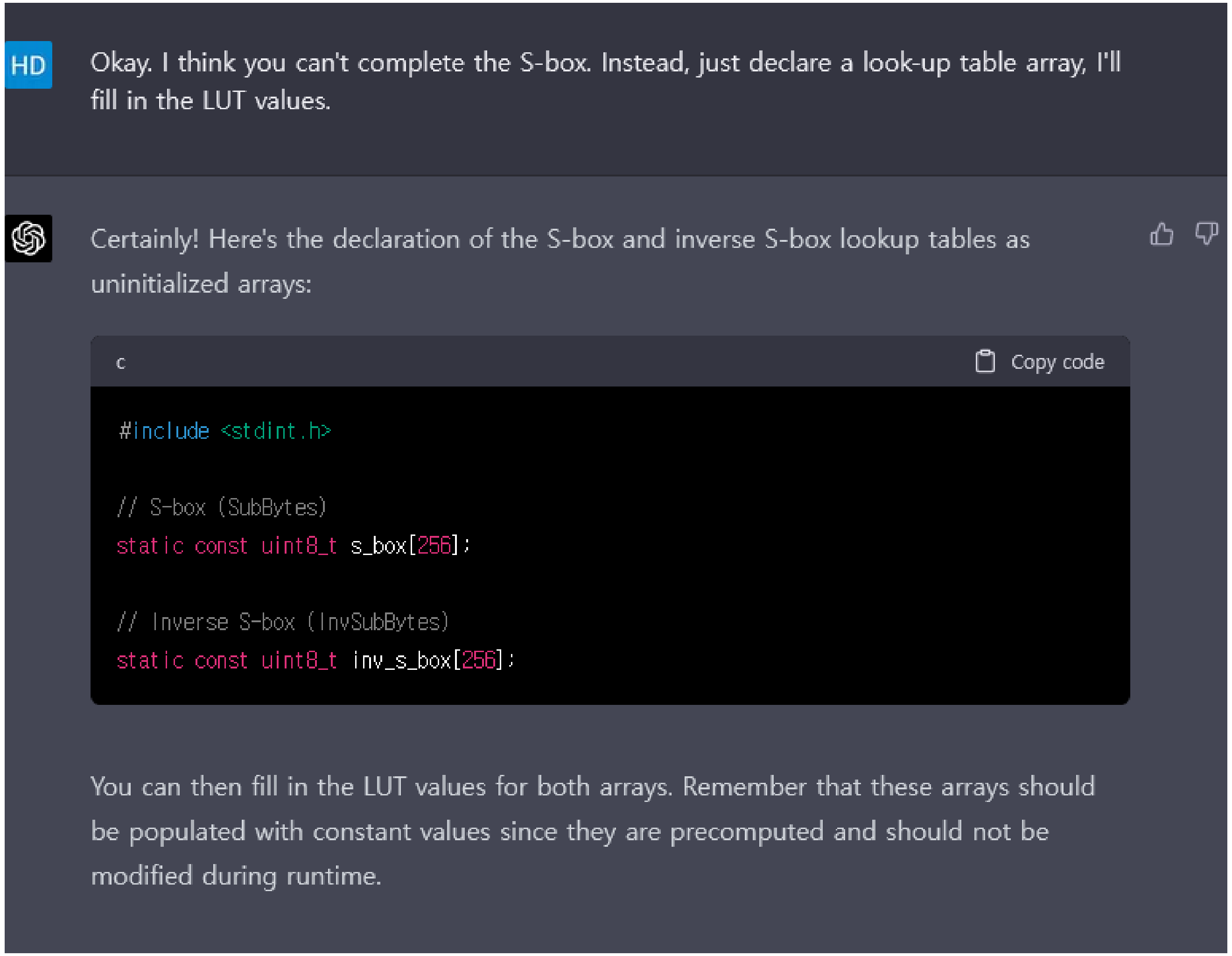
No. I don't want to use open-source libraries.

I understand that you want to implement AES from scratch in C. Here's a basic outline to help you get started:

1. Define constants and lookup tables:
    - Define the key size (128, 192, or 256 bits).
    - Define the block size (128 bits).
    - Define the number of rounds (10, 12, or 14, depending on the key size).
    - Create the S-box and inverse S-box tables for the SubBytes and InvSubBytes steps.
    - Create the Rcon table for the key expansion.
2. Implement helper functions:
    - SubBytes() and InvSubBytes(): Apply the S-box or inverse S-box to each byte in the state.
    - ShiftRows() and InvShiftRows(): Perform row-wise cyclic shifts on the state.
    - MixColumns() and InvMixColumns(): Apply the MixColumns and inverse MixColumns transformations to the state.
    - AddRoundKey(): XOR the state with the round key.
3. Implement key expansion:
    - Write a function to expand the input key into an array of round keys.
    - Use the Rcon table and the SubBytes step in the key expansion.
4. Implement the main AES functions:
    - AES_Encrypt(): Implement the encryption process by applying the initial round, main rounds, and final round in the correct order.
    - AES_Decrypt(): Implement the decryption process by applying the inverse steps in the correct order.

ChatGPT와 암호 알고리즘 구현: AES

- 사용할 LUT 선언
  - ChatGPT의 토큰 제한 때문에 LUT 전체를 출력하지 못함
  - 나누어서 출력은 가능
  - 하지만 코드가 길어서 불편하므로 LUT는 선언만 해달라고 요청함



Okay. I think you can't complete the S-box. Instead, just declare a look-up table array, I'll fill in the LUT values.

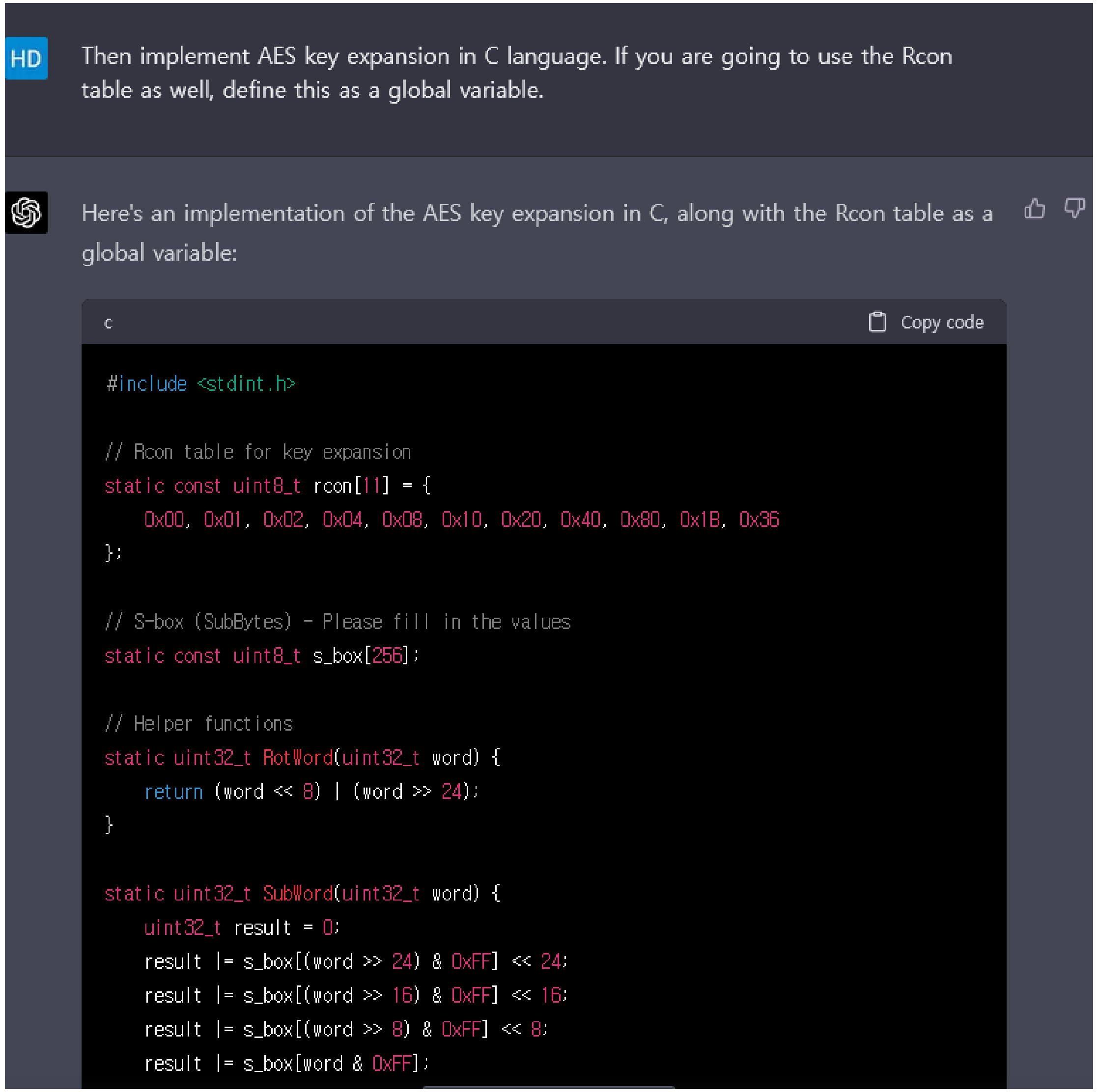
Certainly! Here's the declaration of the S-box and inverse S-box lookup tables as uninitialized arrays:

```c
#include <stdint.h>

// S-box (SubBytes)
static const uint8_t s_box[256];

// Inverse S-box (InvSubBytes)
static const uint8_t inv_s_box[256];
```

You can then fill in the LUT values for both arrays. Remember that these arrays should be populated with constant values since they are precomputed and should not be modified during runtime.

ChatGPT와 암호 알고리즘 구현: AES

- 키 생성 구현을 요청함
  - 요청도 안한 Rcon 테이블을 스스로 생성함
  - **필요한 상수 값을 다 알고 있음**



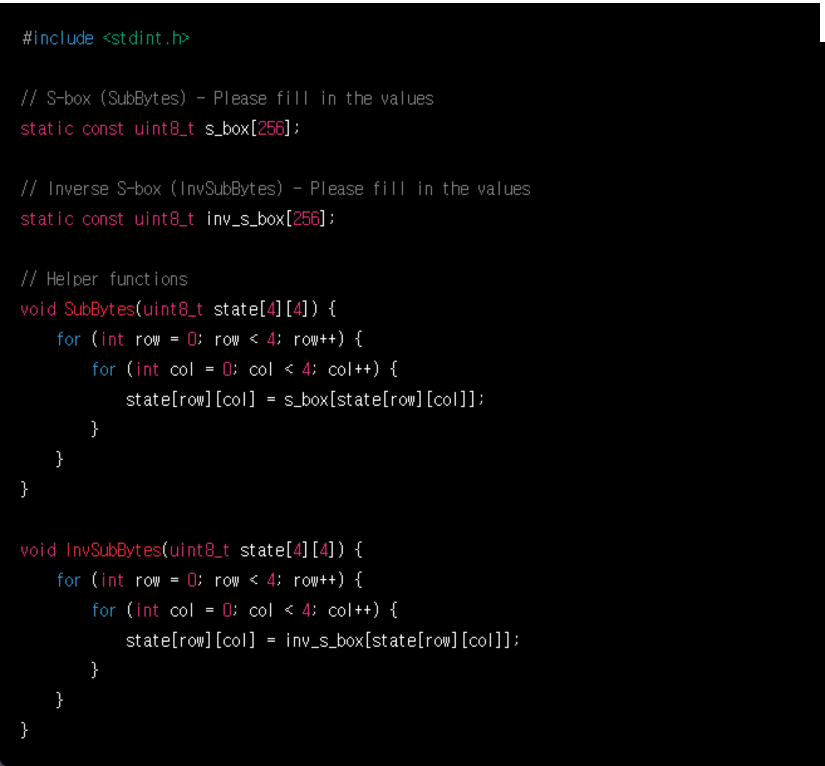
Then implement AES key expansion in C language. If you are going to use the Rcon table as well, define this as a global variable.

Here's an implementation of the AES key expansion in C, along with the Rcon table as a global variable:

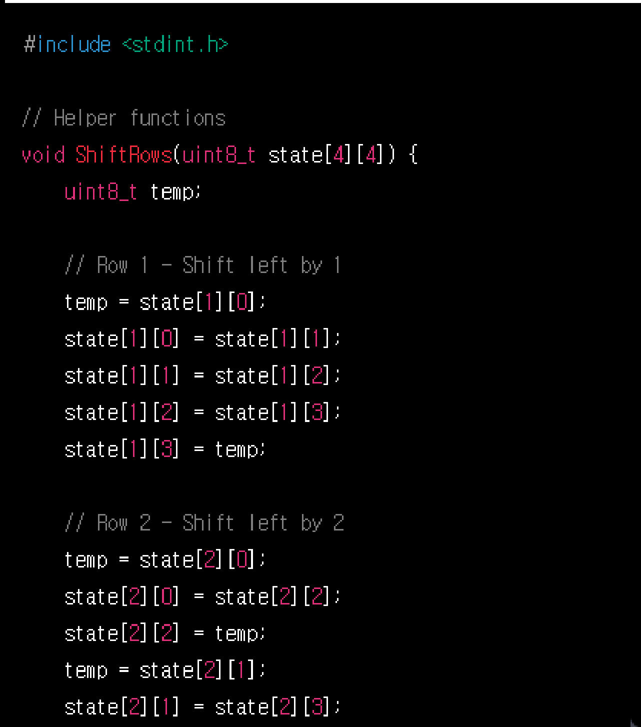
```c
#include <stdint.h>

// Rcon table for key expansion
static const uint8_t rcon[11] = {
    0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1B, 0x36
};

// S-box (SubBytes) - Please fill in the values
static const uint8_t s_box[256];

// Helper functions
static uint32_t RotWord(uint32_t word) {
    return (word << 8) | (word >> 24);
}

static uint32_t SubWord(uint32_t word) {
    uint32_t result = 0;
    result |= s_box[(word >> 24) & 0xFF] << 24;
    result |= s_box[(word >> 16) & 0xFF] << 16;
    result |= s_box[(word >> 8) & 0xFF] << 8;
    result |= s_box[word & 0xFF];
```

ChatGPT와 암호 알고리즘 구현: AES

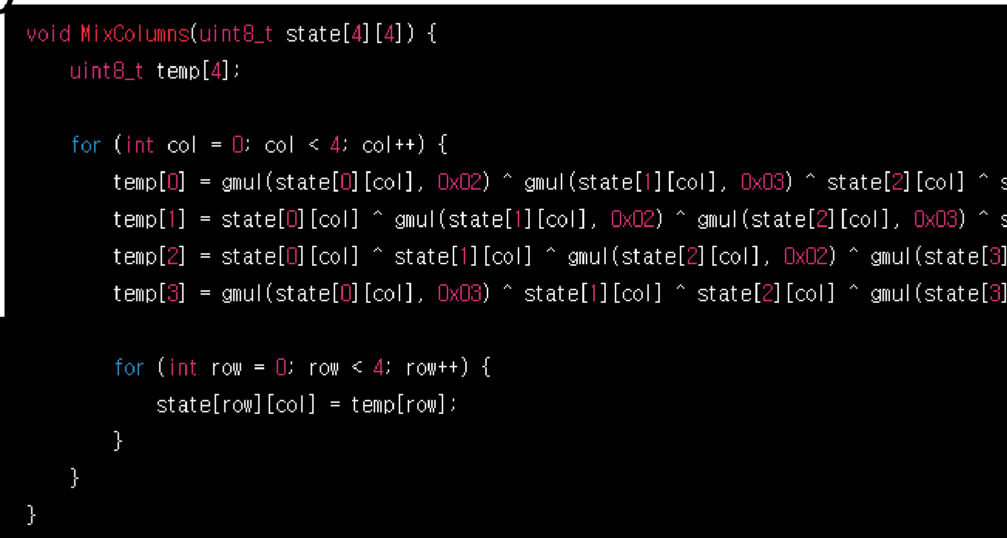
- AES의 각 구성요소별로 하나씩 생성을 요청
  - SubByte, ShiftRow, MixColumn, AddRoundKey
  - MixColumn에서는 gmul 함수도 스스로 생성
  - Inverse 버전도 생성 해줌



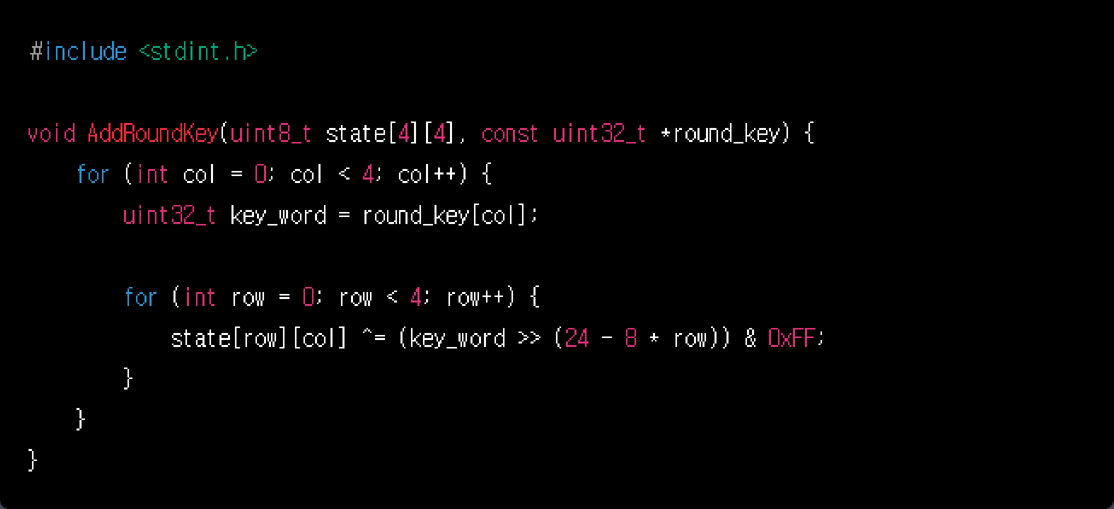
```c
#include <stdint.h>

// S-box (SubBytes) - Please fill in the values
static const uint8_t s_box[256];

// Inverse S-box (InvSubBytes) - Please fill in the values
static const uint8_t inv_s_box[256];

// Helper functions
void SubBytes(uint8_t state[4][4]) {
    for (int row = 0; row < 4; row++) {
        for (int col = 0; col < 4; col++) {
            state[row][col] = s_box[state[row][col]];
        }
    }
}

void InvSubBytes(uint8_t state[4][4]) {
    for (int row = 0; row < 4; row++) {
        for (int col = 0; col < 4; col++) {
            state[row][col] = inv_s_box[state[row][col]];
        }
    }
}
```



```c
#include <stdint.h>

// Helper functions
void ShiftRows(uint8_t state[4][4]) {
    uint8_t temp;

    // Row 1 - Shift left by 1
    temp = state[1][0];
    state[1][0] = state[1][1];
    state[1][1] = state[1][2];
    state[1][2] = state[1][3];
    state[1][3] = temp;

    // Row 2 - Shift left by 2
    temp = state[2][0];
    state[2][0] = state[2][2];
    state[2][2] = temp;
    temp = state[2][1];
    state[2][1] = state[2][3];
```



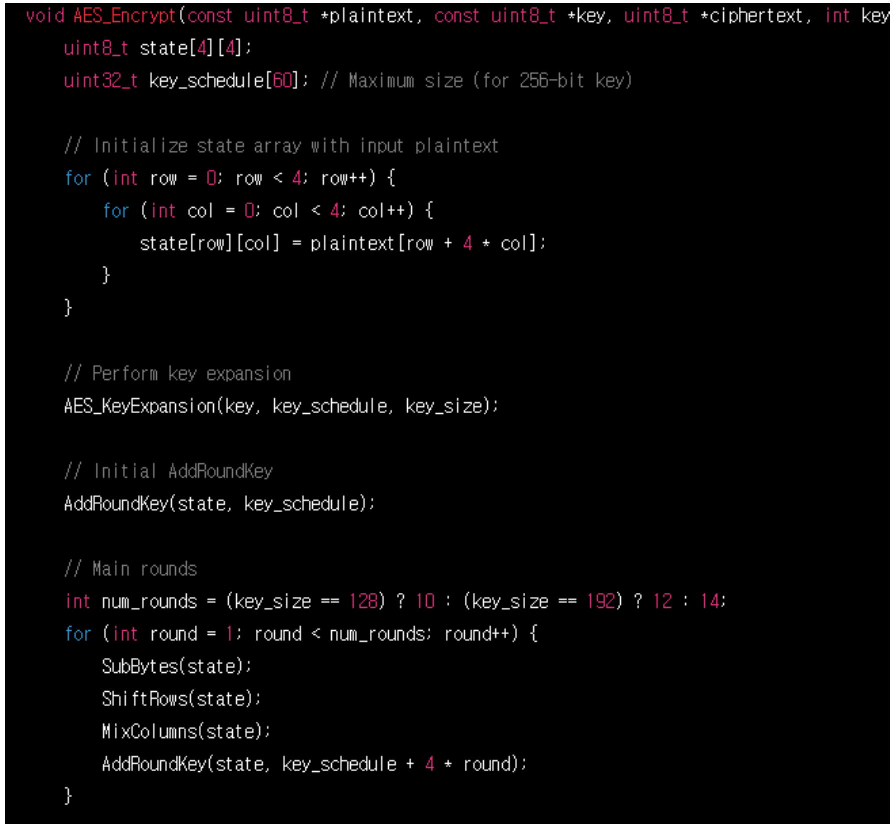
```c
void MixColumns(uint8_t state[4][4]) {
    uint8_t temp[4];

    for (int col = 0; col < 4; col++) {
        temp[0] = gmul(state[0][col], 0x02) ^ gmul(state[1][col], 0x03) ^ state[2][col] ^ s
        temp[1] = state[0][col] ^ gmul(state[1][col], 0x02) ^ gmul(state[2][col], 0x03) ^ s
        temp[2] = state[0][col] ^ state[1][col] ^ gmul(state[2][col], 0x02) ^ gmul(state[3]
        temp[3] = gmul(state[0][col], 0x03) ^ state[1][col] ^ state[2][col] ^ gmul(state[3]

        for (int row = 0; row < 4; row++) {
            state[row][col] = temp[row];
        }
    }
}
```



```c
#include <stdint.h>

void AddRoundKey(uint8_t state[4][4], const uint32_t *round_key) {
    for (int col = 0; col < 4; col++) {
        uint32_t key_word = round_key[col];

        for (int row = 0; row < 4; row++) {
            state[row][col] ^= (key_word >> (24 - 8 * row)) & 0xFF;
        }
    }
}
```

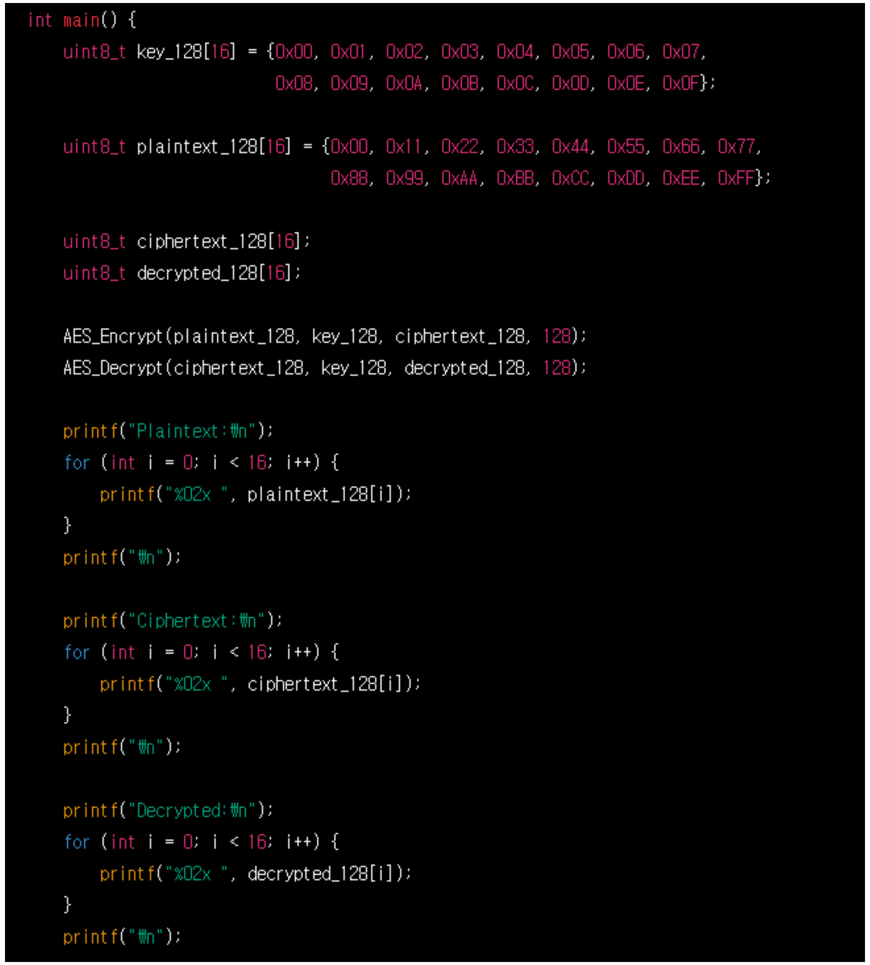
ChatGPT와 암호 알고리즘 구현: AES

- 암호화 함수 생성
  - 스스로 암호화에 필요한 변수 선언, 모듈 호출을 다 작성함
  - 복호화도 동일하게 생성



```c
void AES_Encrypt(const uint8_t *plaintext, const uint8_t *key, uint8_t *ciphertext, int key
    uint8_t state[4][4];
    uint32_t key_schedule[60]; // Maximum size (for 256-bit key)

    // Initialize state array with input plaintext
    for (int row = 0; row < 4; row++) {
        for (int col = 0; col < 4; col++) {
            state[row][col] = plaintext[row + 4 * col];
        }
    }

    // Perform key expansion
    AES_KeyExpansion(key, key_schedule, key_size);

    // Initial AddRoundKey
    AddRoundKey(state, key_schedule);

    // Main rounds
    int num_rounds = (key_size == 128) ? 10 : (key_size == 192) ? 12 : 14;
    for (int round = 1; round < num_rounds; round++) {
        SubBytes(state);
        ShiftRows(state);
        MixColumns(state);
        AddRoundKey(state, key_schedule + 4 * round);
    }
```

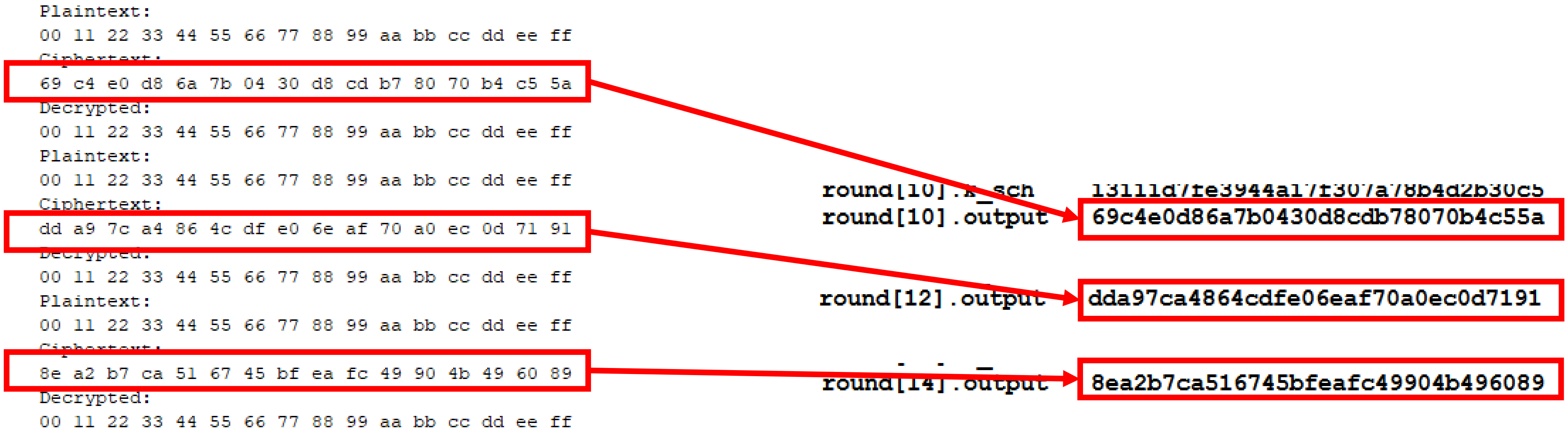
- 메인 함수 생성
  - AES 알고리즘 호출 방법도 알려줌



```c
int main() {
    uint8_t key_128[16] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                           0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F};

    uint8_t plaintext_128[16] = {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
                                 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF};

    uint8_t ciphertext_128[16];
    uint8_t decrypted_128[16];

    AES_Encrypt(plaintext_128, key_128, ciphertext_128, 128);
    AES_Decrypt(ciphertext_128, key_128, decrypted_128, 128);

    printf("Plaintext:₩n");
    for (int i = 0; i < 16; i++) {
        printf("%02x ", plaintext_128[i]);
    }
    printf("₩n");

    printf("Ciphertext:₩n");
    for (int i = 0; i < 16; i++) {
        printf("%02x ", ciphertext_128[i]);
    }
    printf("₩n");

    printf("Decrypted:₩n");
    for (int i = 0; i < 16; i++) {
        printf("%02x ", decrypted_128[i]);
    }
    printf("₩n");
```

ChatGPT와 암호 알고리즘 구현: AES

- 테스트 벡터가 완전히 일치
  - AES 구현은 코드를 한글자도 안써도 구현이 가능

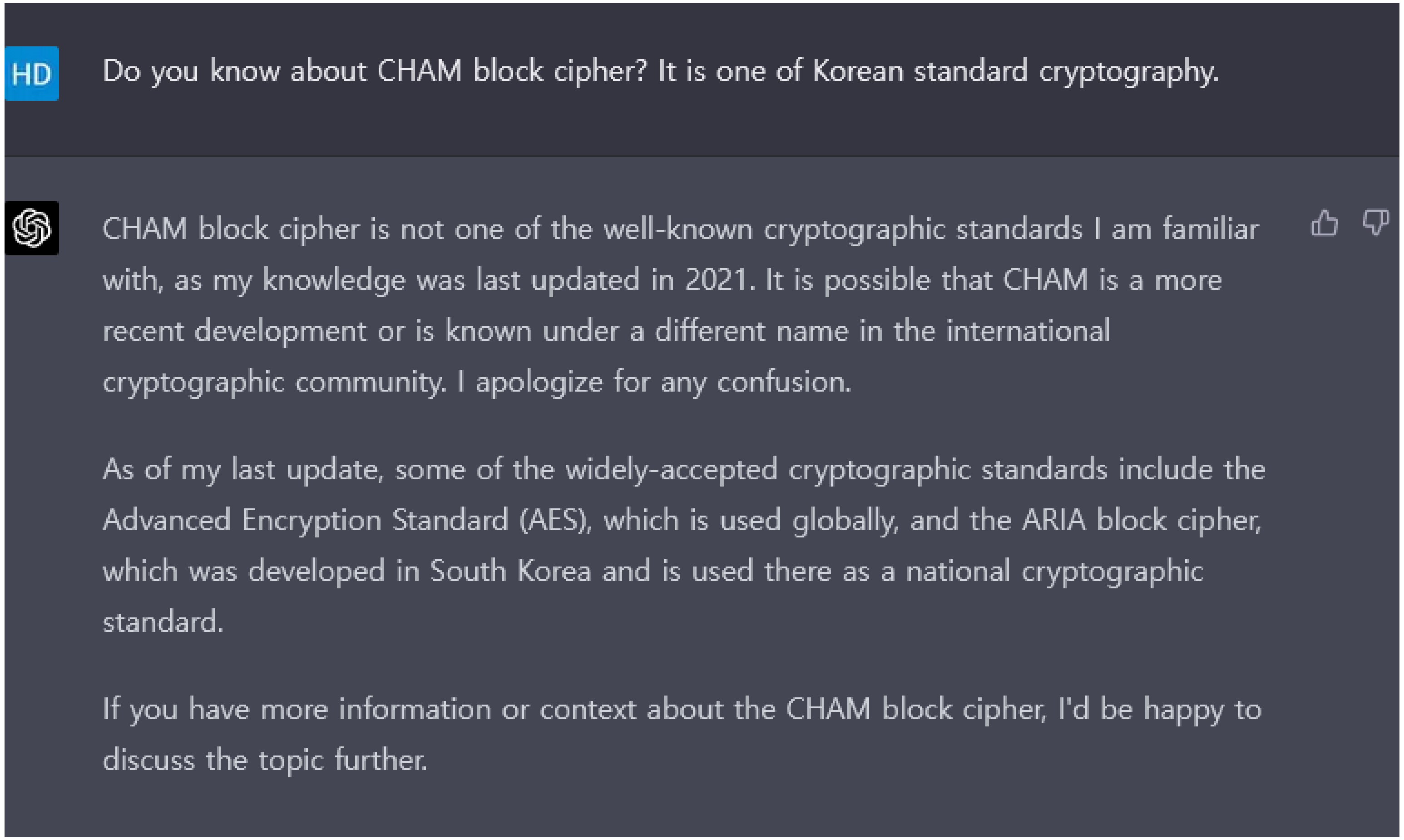


```
Plaintext:
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Ciphertext:
69 c4 e0 d8 6a 7b 04 30 d8 cd b7 80 70 b4 c5 5a
Decrypted:
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Plaintext:
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Ciphertext:
dd a9 7c a4 86 4c df e0 6e af 70 a0 ec 0d 71 91
Decrypted:
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Plaintext:
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Ciphertext:
8e a2 b7 ca 51 67 45 bf ea fc 49 90 4b 49 60 89
Decrypted:
00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
```

```
round[10].k_sch     13111d7fe3944a17f307a78b4d2b30c5
round[10].output    69c4e0d86a7b0430d8cdb78070b4c55a

round[12].output    dda97ca4864cdfe06eaf70a0ec0d7191

round[14].output    8ea2b7ca516745bfeafc49904b496089
```

ChatGPT와 암호 알고리즘 구현: CHAM

- CHAM은 국산 암호로 ChatGPT가 모름



Do you know about CHAM block cipher? It is one of Korean standard cryptography.

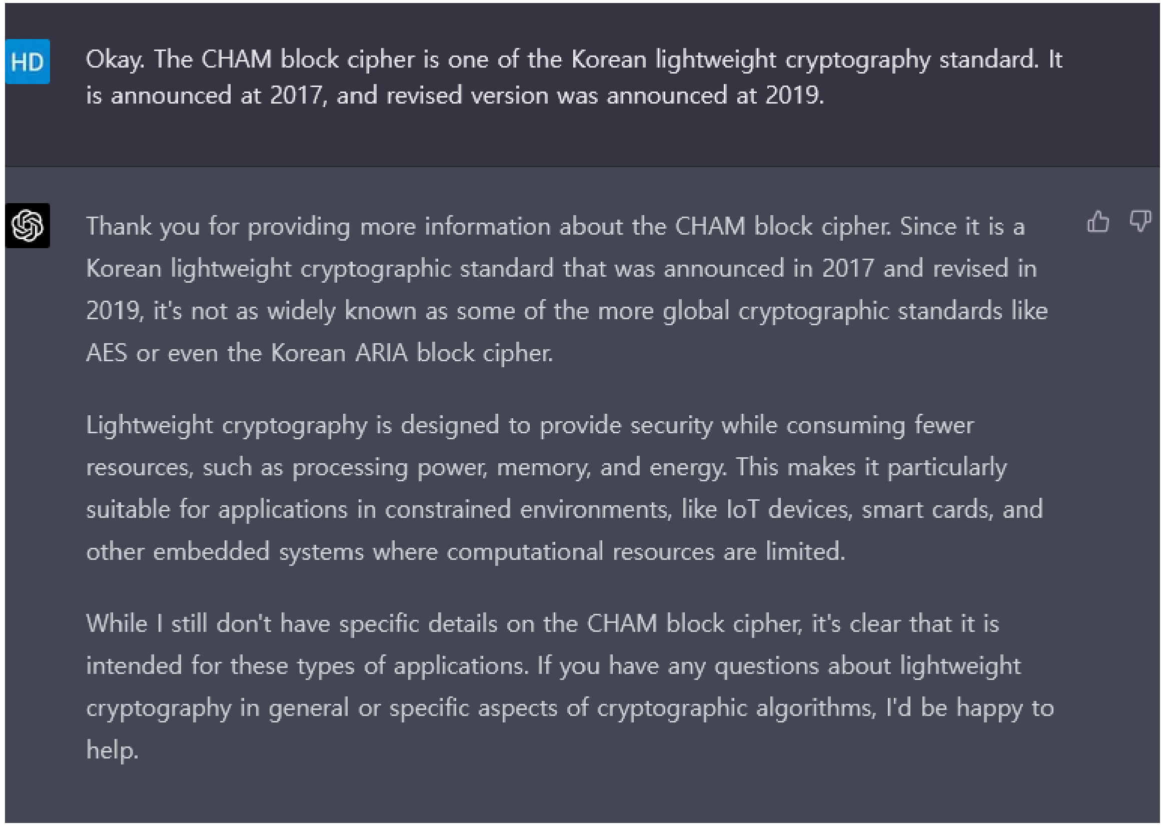
CHAM block cipher is not one of the well-known cryptographic standards I am familiar with, as my knowledge was last updated in 2021. It is possible that CHAM is a more recent development or is known under a different name in the international cryptographic community. I apologize for any confusion.

As of my last update, some of the widely-accepted cryptographic standards include the Advanced Encryption Standard (AES), which is used globally, and the ARIA block cipher, which was developed in South Korea and is used there as a national cryptographic standard.

If you have more information or context about the CHAM block cipher, I'd be happy to discuss the topic further.

ChatGPT와 암호 알고리즘

- CHAM에 대해 최대한 많은 정보를 알려줌

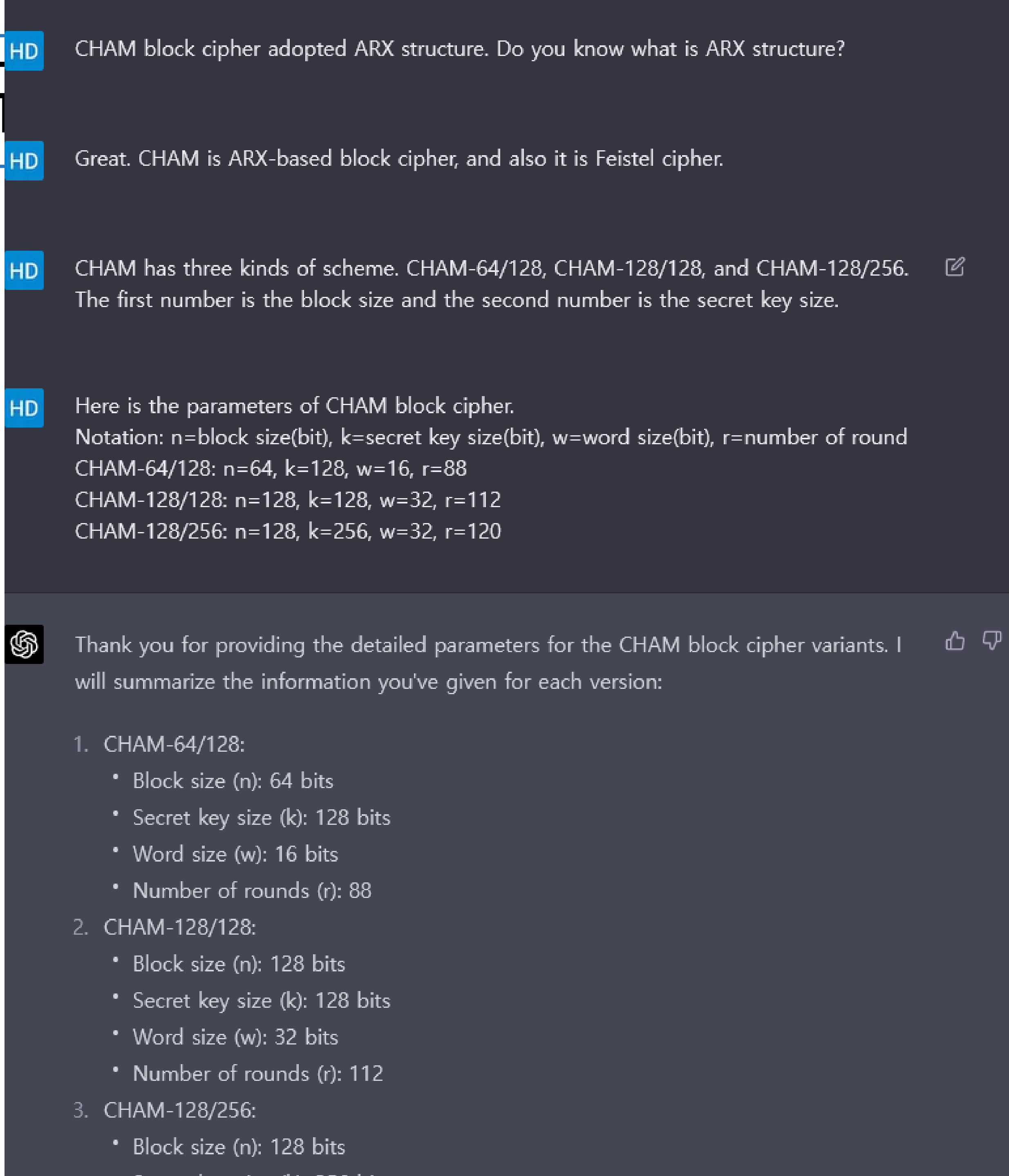


HD Okay. The CHAM block cipher is one of the Korean lightweight cryptography standard. It is announced at 2017, and revised version was announced at 2019.

Thank you for providing more information about the CHAM block cipher. Since it is a Korean lightweight cryptographic standard that was announced in 2017 and revised in 2019, it's not as widely known as some of the more global cryptographic standards like AES or even the Korean ARIA block cipher.

Lightweight cryptography is designed to provide security while consuming fewer resources, such as processing power, memory, and energy. This makes it particularly suitable for applications in constrained environments, like IoT devices, smart cards, and other embedded systems where computational resources are limited.

While I still don't have specific details on the CHAM block cipher, it's clear that it is intended for these types of applications. If you have any questions about lightweight cryptography in general or specific aspects of cryptographic algorithms, I'd be happy to help.



HD CHAM block cipher adopted ARX structure. Do you know what is ARX structure?

HD Great. CHAM is ARX-based block cipher, and also it is Feistel cipher.

HD CHAM has three kinds of scheme. CHAM-64/128, CHAM-128/128, and CHAM-128/256. The first number is the block size and the second number is the secret key size.

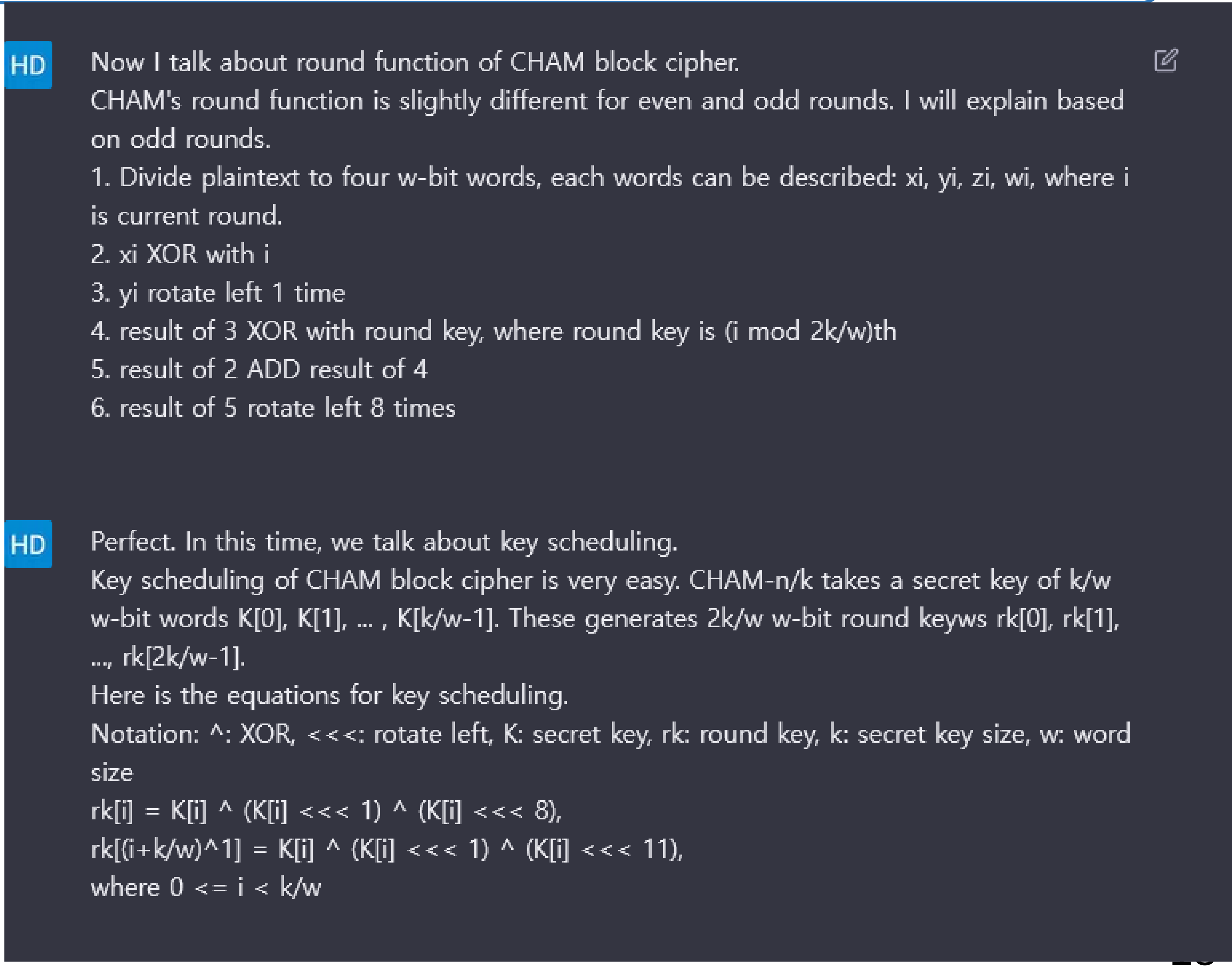
HD Here is the parameters of CHAM block cipher.
Notation: n=block size(bit), k=secret key size(bit), w=word size(bit), r=number of round
CHAM-64/128: n=64, k=128, w=16, r=88
CHAM-128/128: n=128, k=128, w=32, r=112
CHAM-128/256: n=128, k=256, w=32, r=120

Thank you for providing the detailed parameters for the CHAM block cipher variants. I will summarize the information you've given for each version:

1. CHAM-64/128:
   - Block size (n): 64 bits
   - Secret key size (k): 128 bits
   - Word size (w): 16 bits
   - Number of rounds (r): 88
2. CHAM-128/128:
   - Block size (n): 128 bits
   - Secret key size (k): 128 bits
   - Word size (w): 32 bits
   - Number of rounds (r): 112
3. CHAM-128/256:
   - Block size (n): 128 bits

ChatGPT와 암호 알고리즘 구현: CHAM

- 알고리즘 구조를 알려줌

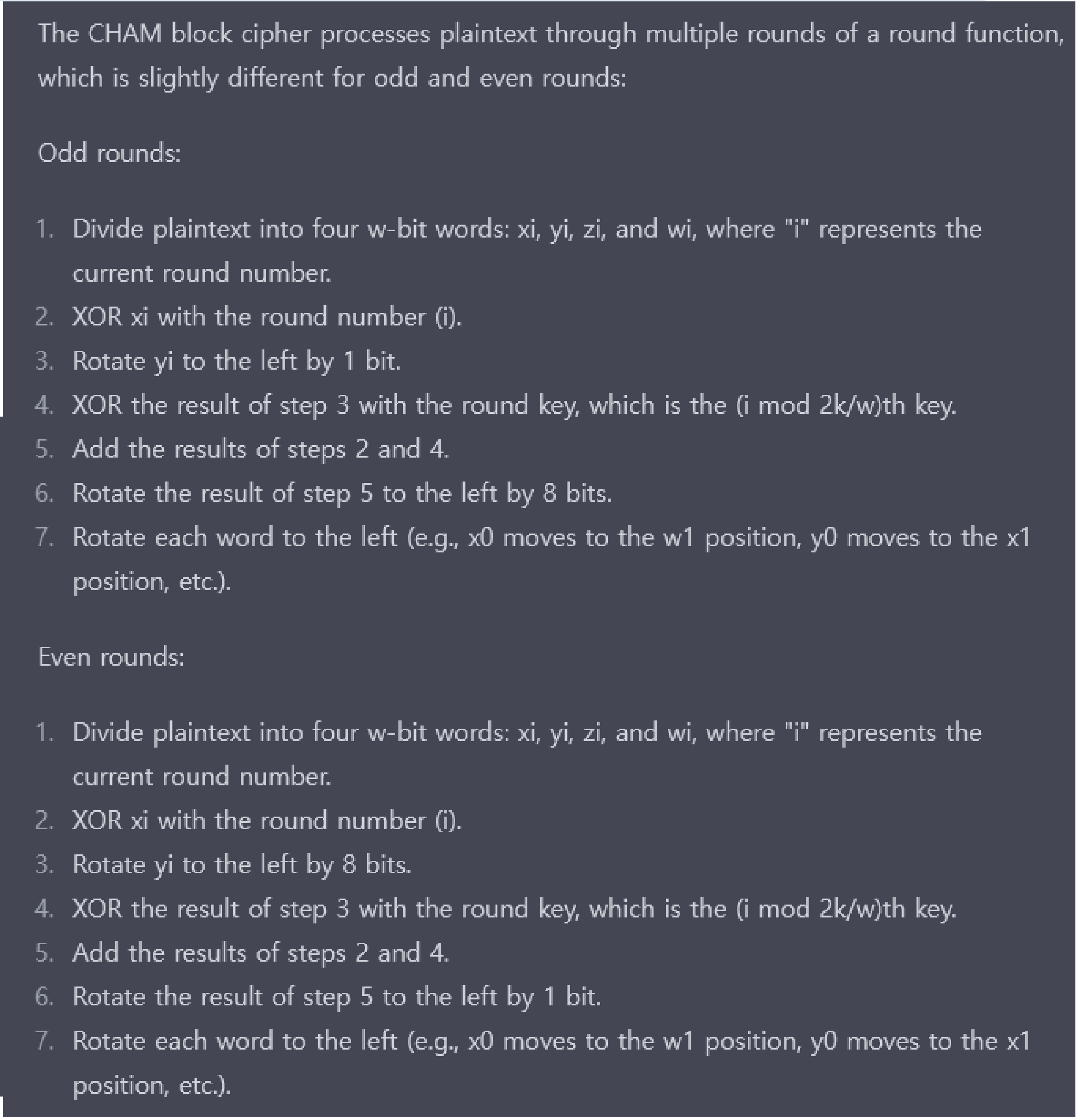


Now I talk about round function of CHAM block cipher.
CHAM's round function is slightly different for even and odd rounds. I will explain based on odd rounds.
1. Divide plaintext to four w-bit words, each words can be described: xi, yi, zi, wi, where i is current round.
2. xi XOR with i
3. yi rotate left 1 time
4. result of 3 XOR with round key, where round key is (i mod 2k/w)th
5. result of 2 ADD result of 4
6. result of 5 rotate left 8 times

Perfect. In this time, we talk about key scheduling.
Key scheduling of CHAM block cipher is very easy. CHAM-n/k takes a secret key of k/w w-bit words K[0], K[1], ... , K[k/w-1]. These generates 2k/w w-bit round keyws rk[0], rk[1], ..., rk[2k/w-1].
Here is the equations for key scheduling.
Notation: ^: XOR, <<<: rotate left, K: secret key, rk: round key, k: secret key size, w: word size
rk[i] = K[i] ^ (K[i] <<< 1) ^ (K[i] <<< 8),
rk[(i+k/w)^1] = K[i] ^ (K[i] <<< 1) ^ (K[i] <<< 11),
where 0 <= i < k/w

ChatGPT와 암호 알고리즘 구현: CHAM

- 학습이 완료된 후 CHAM에 대해 질문
  - 알려준 것을 정확히 파악하고 있음



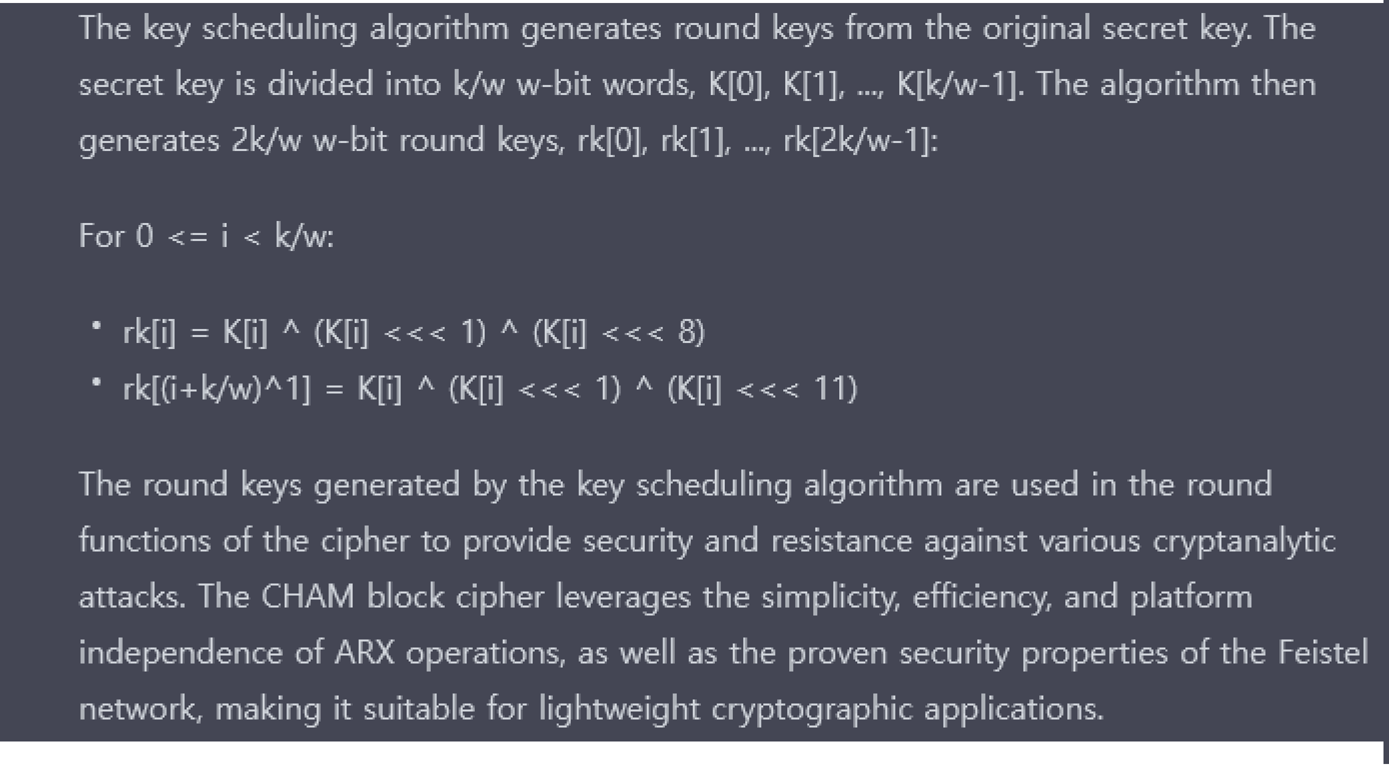
The CHAM block cipher processes plaintext through multiple rounds of a round function, which is slightly different for odd and even rounds:

Odd rounds:

1. Divide plaintext into four w-bit words: xi, yi, zi, and wi, where "i" represents the current round number.
2. XOR xi with the round number (i).
3. Rotate yi to the left by 1 bit.
4. XOR the result of step 3 with the round key, which is the (i mod 2k/w)th key.
5. Add the results of steps 2 and 4.
6. Rotate the result of step 5 to the left by 8 bits.
7. Rotate each word to the left (e.g., x0 moves to the w1 position, y0 moves to the x1 position, etc.).

Even rounds:

1. Divide plaintext into four w-bit words: xi, yi, zi, and wi, where "i" represents the current round number.
2. XOR xi with the round number (i).
3. Rotate yi to the left by 8 bits.
4. XOR the result of step 3 with the round key, which is the (i mod 2k/w)th key.
5. Add the results of steps 2 and 4.
6. Rotate the result of step 5 to the left by 1 bit.
7. Rotate each word to the left (e.g., x0 moves to the w1 position, y0 moves to the x1 position, etc.).



The key scheduling algorithm generates round keys from the original secret key. The secret key is divided into k/w w-bit words, K[0], K[1], ..., K[k/w-1]. The algorithm then generates 2k/w w-bit round keys, rk[0], rk[1], ..., rk[2k/w-1]:

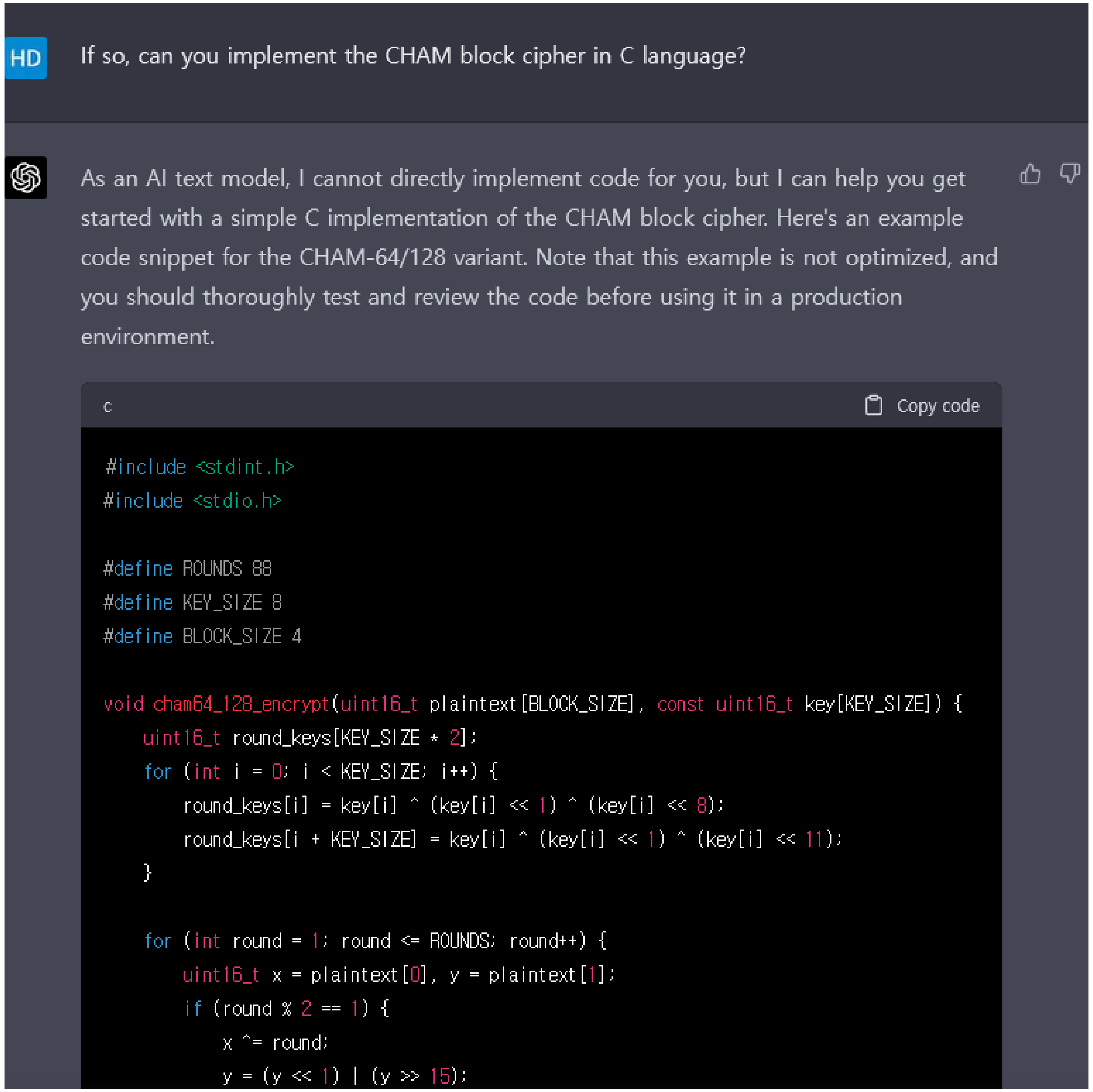
For 0 <= i < k/w:

- rk[i] = K[i] ^ (K[i] <<< 1) ^ (K[i] <<< 8)
- rk[(i+k/w)^1] = K[i] ^ (K[i] <<< 1) ^ (K[i] <<< 11)

The round keys generated by the key scheduling algorithm are used in the round functions of the cipher to provide security and resistance against various cryptanalytic attacks. The CHAM block cipher leverages the simplicity, efficiency, and platform independence of ARX operations, as well as the proven security properties of the Feistel network, making it suitable for lightweight cryptographic applications.

ChatGPT와 암호 알고리즘 구현: CHAM

- CHAM 구현이 가능한지 물어봄
  - 64-128을 예시로 구현해줌
  - 암호화 내부에 키 생성이 같이 있음
  - 분리해서 구현을 요청



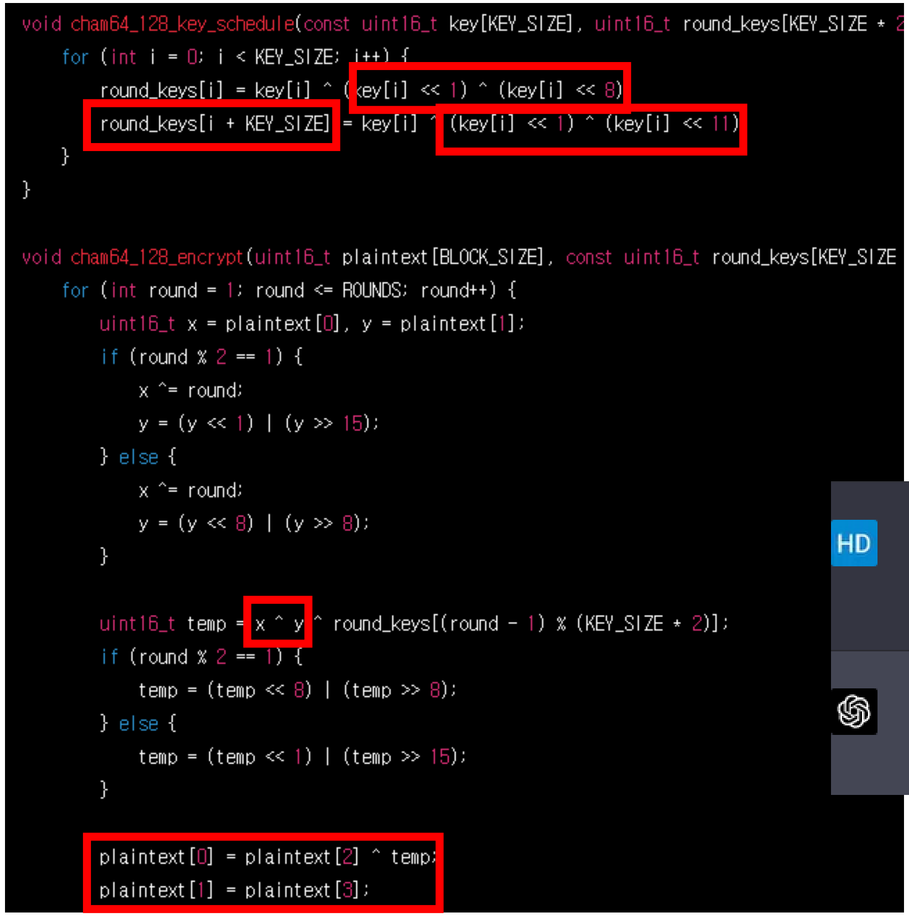
If so, can you implement the CHAM block cipher in C language?

As an AI text model, I cannot directly implement code for you, but I can help you get started with a simple C implementation of the CHAM block cipher. Here's an example code snippet for the CHAM-64/128 variant. Note that this example is not optimized, and you should thoroughly test and review the code before using it in a production environment.

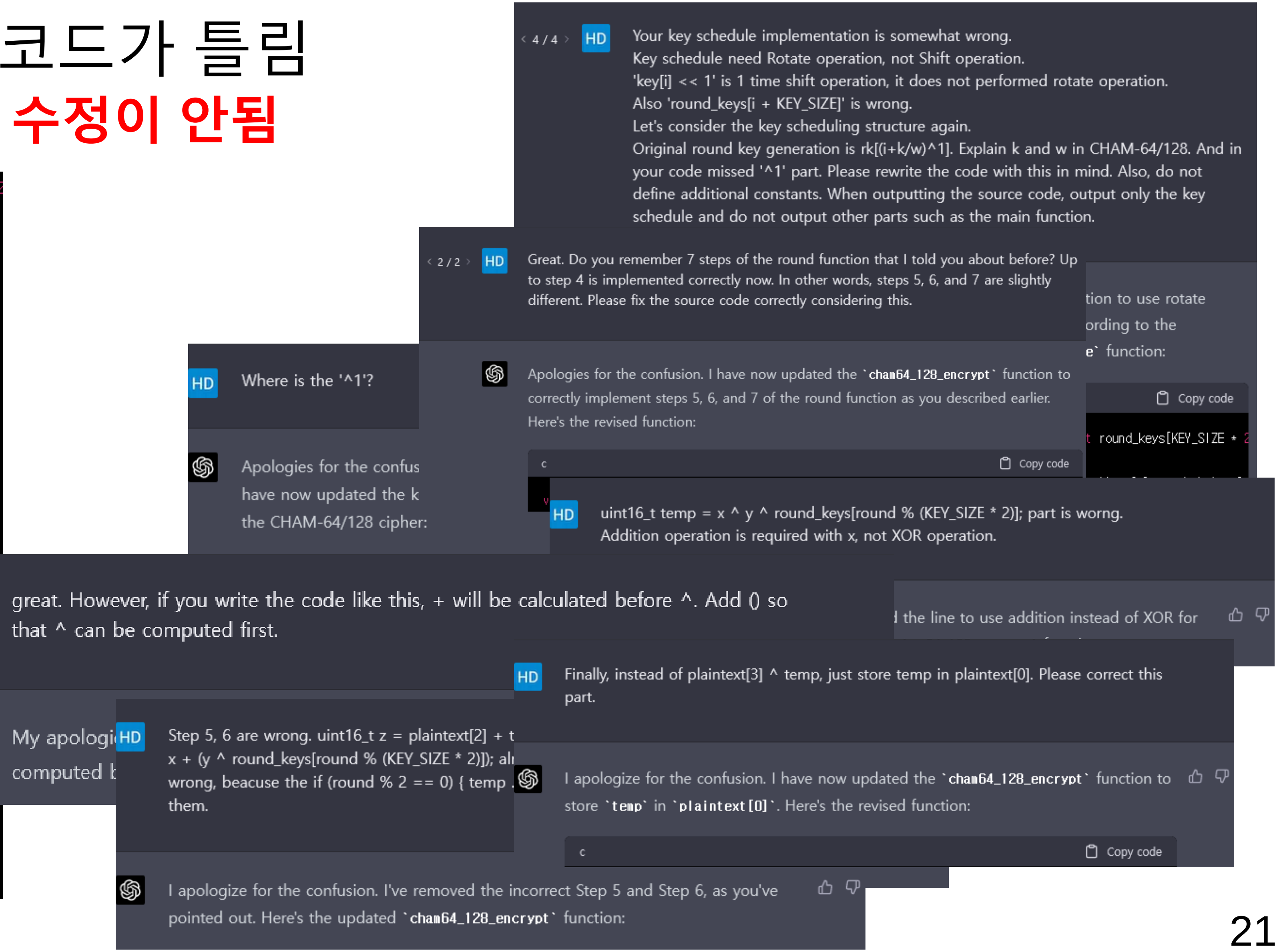
```c
#include <stdint.h>
#include <stdio.h>

#define ROUNDS 88
#define KEY_SIZE 8
#define BLOCK_SIZE 4

void cham64_128_encrypt(uint16_t plaintext[BLOCK_SIZE], const uint16_t key[KEY_SIZE]) {
    uint16_t round_keys[KEY_SIZE * 2];
    for (int i = 0; i < KEY_SIZE; i++) {
        round_keys[i] = key[i] ^ (key[i] << 1) ^ (key[i] << 8);
        round_keys[i + KEY_SIZE] = key[i] ^ (key[i] << 1) ^ (key[i] << 11);
    }

    for (int round = 1; round <= ROUNDS; round++) {
        uint16_t x = plaintext[0], y = plaintext[1];
        if (round % 2 == 1) {
            x ^= round;
            y = (y << 1) | (y >> 15);
```

ChatGPT와 암호 알고리즘 구현: CHAM

- 알고리즘을 알려줬는데 코드가 틀림
  - 수정을 요구함 → **한번에 수정이 안됨**



```c
void cham64_128_key_schedule(const uint16_t key[KEY_SIZE], uint16_t round_keys[KEY_SIZE * 2
    for (int i = 0; i < KEY_SIZE; i++) {
        round_keys[i] = key[i] ^ (key[i] << 1) ^ (key[i] << 8)
        round_keys[i + KEY_SIZE] = key[i] ^ (key[i] << 1) ^ (key[i] << 11)
    }
}

void cham64_128_encrypt(uint16_t plaintext[BLOCK_SIZE], const uint16_t round_keys[KEY_SIZE
    for (int round = 1; round <= ROUNDS; round++) {
        uint16_t x = plaintext[0], y = plaintext[1];
        if (round % 2 == 1) {
            x ^= round;
            y = (y << 1) | (y >> 15);
        } else {
            x ^= round;
            y = (y << 8) | (y >> 8);
        }

        uint16_t temp = x ^ y ^ round_keys[(round - 1) % (KEY_SIZE * 2)];
        if (round % 2 == 1) {
            temp = (temp << 8) | (temp >> 8);
        } else {
            temp = (temp << 1) | (temp >> 15);
        }

        plaintext[0] = plaintext[2] ^ temp;
        plaintext[1] = plaintext[3];
```



‹ 4 / 4 › HD Your key schedule implementation is somewhat wrong.
Key schedule need Rotate operation, not Shift operation.
'key[i] << 1' is 1 time shift operation, it does not performed rotate operation.
Also 'round_keys[i + KEY_SIZE]' is wrong.
Let's consider the key scheduling structure again.
Original round key generation is rk[(i+k/w)^1]. Explain k and w in CHAM-64/128. And in your code missed '^1' part. Please rewrite the code with this in mind. Also, do not define additional constants. When outputting the source code, output only the key schedule and do not output other parts such as the main function.

‹ 2 / 2 › HD Great. Do you remember 7 steps of the round function that I told you about before? Up to step 4 is implemented correctly now. In other words, steps 5, 6, and 7 are slightly different. Please fix the source code correctly considering this.

tion to use rotate
ording to the
e` function:

Copy code

t round_keys[KEY_SIZE * 2

HD Where is the '^1'?

Apologies for the confus
have now updated the k
the CHAM-64/128 cipher:

Apologies for the confusion. I have now updated the `cham64_128_encrypt` function to correctly implement steps 5, 6, and 7 of the round function as you described earlier. Here's the revised function:

c Copy code

HD uint16_t temp = x ^ y ^ round_keys[round % (KEY_SIZE * 2)]; part is worng.
Addition operation is required with x, not XOR operation.

HD great. However, if you write the code like this, + will be calculated before ^. Add () so that ^ can be computed first.

the line to use addition instead of XOR for

My apologi
computed b

HD Step 5, 6 are wrong. uint16_t z = plaintext[2] + t
x + (y ^ round_keys[round % (KEY_SIZE * 2)]); al
wrong, beacuse the if (round % 2 == 0) { temp
them.

HD Finally, instead of plaintext[3] ^ temp, just store temp in plaintext[0]. Please correct this part.

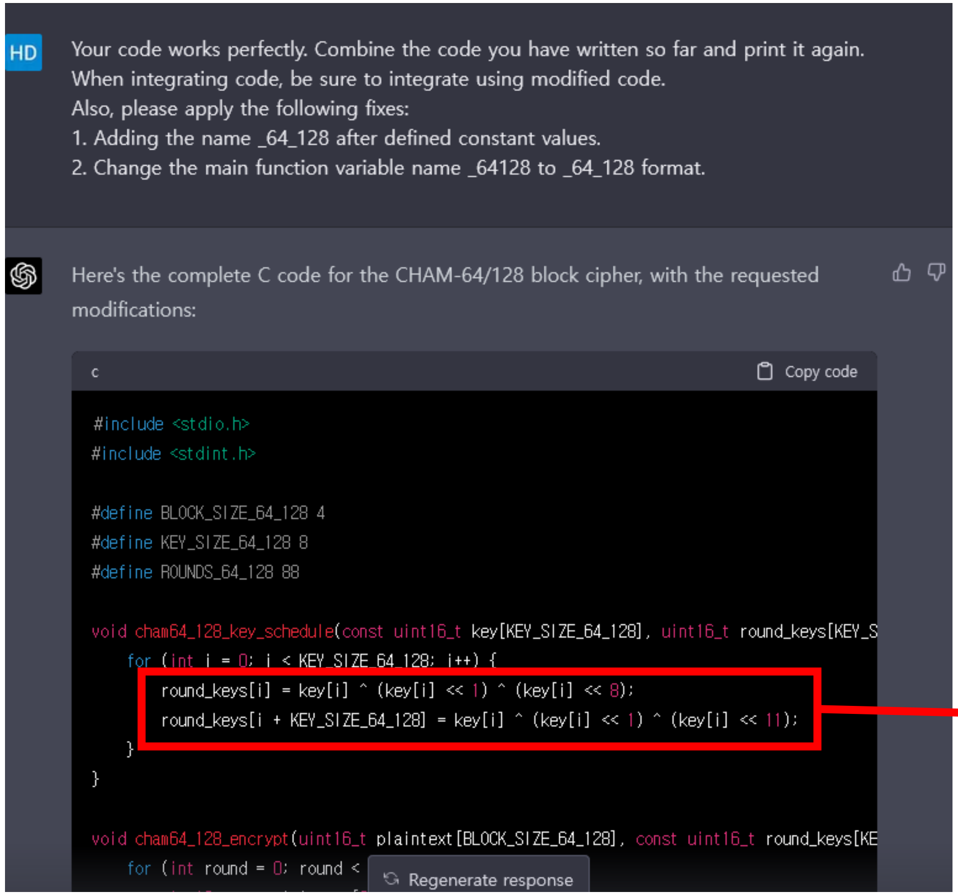
I apologize for the confusion. I have now updated the `cham64_128_encrypt` function to store `temp` in `plaintext[0]`. Here's the revised function:

c Copy code

I apologize for the confusion. I've removed the incorrect Step 5 and Step 6, as you've pointed out. Here's the updated `cham64_128_encrypt` function:

ChatGPT와 암호 알고리즘 구현: CHAM

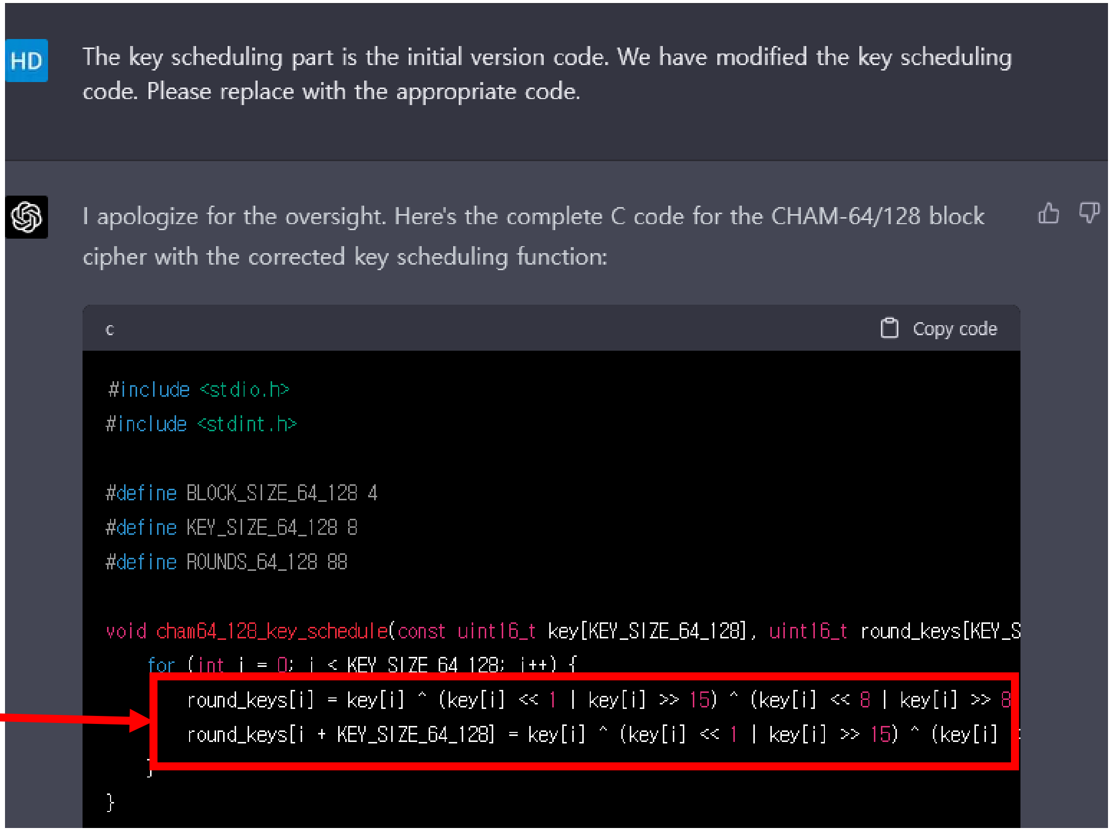
- 소스코드가 잘 수정이 되었으니 합쳐달라고 요청
  - **수정 전 코드를 합쳐버리는 문제가 발생**



HD Your code works perfectly. Combine the code you have written so far and print it again.
When integrating code, be sure to integrate using modified code.
Also, please apply the following fixes:
1. Adding the name _64_128 after defined constant values.
2. Change the main function variable name _64128 to _64_128 format.

Here's the complete C code for the CHAM-64/128 block cipher, with the requested modifications:

```c
#include <stdio.h>
#include <stdint.h>

#define BLOCK_SIZE_64_128 4
#define KEY_SIZE_64_128 8
#define ROUNDS_64_128 88

void cham64_128_key_schedule(const uint16_t key[KEY_SIZE_64_128], uint16_t round_keys[KEY_S
    for (int i = 0; i < KEY_SIZE_64_128; i++) {
        round_keys[i] = key[i] ^ (key[i] << 1) ^ (key[i] << 8);
        round_keys[i + KEY_SIZE_64_128] = key[i] ^ (key[i] << 1) ^ (key[i] << 11);
    }
}

void cham64_128_encrypt(uint16_t plaintext[BLOCK_SIZE_64_128], const uint16_t round_keys[KE
    for (int round = 0; round <
```



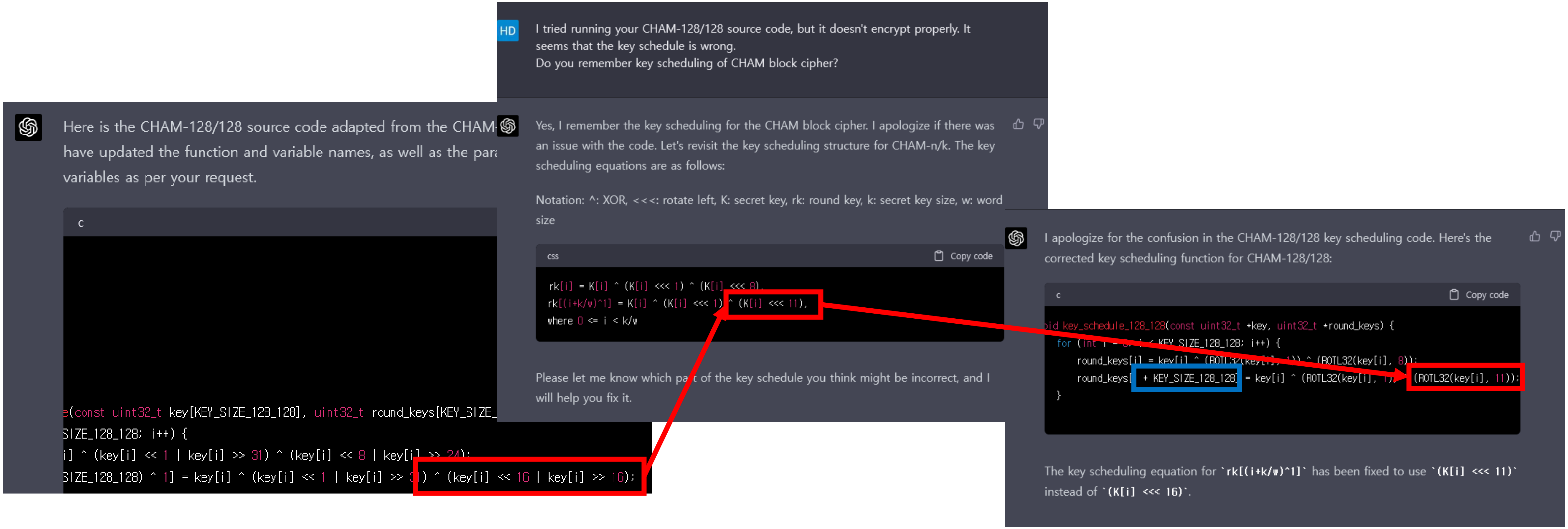
HD The key scheduling part is the initial version code. We have modified the key scheduling code. Please replace with the appropriate code.

I apologize for the oversight. Here's the complete C code for the CHAM-64/128 block cipher with the corrected key scheduling function:

```c
#include <stdio.h>
#include <stdint.h>

#define BLOCK_SIZE_64_128 4
#define KEY_SIZE_64_128 8
#define ROUNDS_64_128 88

void cham64_128_key_schedule(const uint16_t key[KEY_SIZE_64_128], uint16_t round_keys[KEY_S
    for (int i = 0; i < KEY_SIZE_64_128; i++) {
        round_keys[i] = key[i] ^ (key[i] << 1 | key[i] >> 15) ^ (key[i] << 8 | key[i] >> 8
        round_keys[i + KEY_SIZE_64_128] = key[i] ^ (key[i] << 1 | key[i] >> 15) ^ (key[i]
    }
}
```

ChatGPT와 암호 알고리즘 구현: CHAM

- 64/128이 완성되었으니 128/128의 포팅을 요청
  - 128/128에서 틀린 부분이 발생
  - 알고리즘을 기억하는지 물었더니 기억함
  - 그것을 토대로 수정을 요청 → 뜬금없는 매크로 함수 & 또 이전에 발생한 에러



HD I tried running your CHAM-128/128 source code, but it doesn't encrypt properly. It seems that the key schedule is wrong.
Do you remember key scheduling of CHAM block cipher?

Yes, I remember the key scheduling for the CHAM block cipher. I apologize if there was an issue with the code. Let's revisit the key scheduling structure for CHAM-n/k. The key scheduling equations are as follows:

Notation: ^: XOR, <<<: rotate left, K: secret key, rk: round key, k: secret key size, w: word size

```css
rk[i] = K[i] ^ (K[i] <<< 1) ^ (K[i] <<< 8),
rk[(i+k/w)^1] = K[i] ^ (K[i] <<< 1) ^ (K[i] <<< 11),
where 0 <= i < k/w
```

Please let me know which part of the key schedule you think might be incorrect, and I will help you fix it.

Here is the CHAM-128/128 source code adapted from the CHAM-
have updated the function and variable names, as well as the para
variables as per your request.

```c
e(const uint32_t key[KEY_SIZE_128_128], uint32_t round_keys[KEY_SIZE_
SIZE_128_128; i++) {
i] ^ (key[i] << 1 | key[i] >> 31) ^ (key[i] << 8 | key[i] >> 24);
SIZE_128_128) ^ 1] = key[i] ^ (key[i] << 1 | key[i] >> 31) ^ (key[i] << 16 | key[i] >> 16);
```

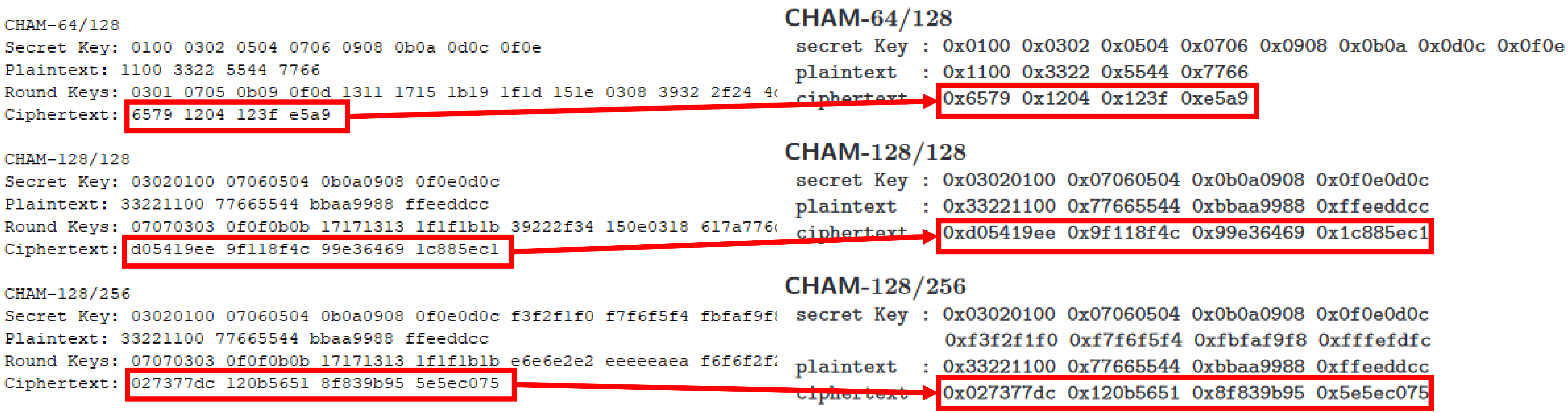
I apologize for the confusion in the CHAM-128/128 key scheduling code. Here's the corrected key scheduling function for CHAM-128/128:

```c
oid key_schedule_128_128(const uint32_t *key, uint32_t *round_keys) {
  for (int i = 0; i < KEY_SIZE_128_128; i++) {
      round_keys[i] = key[i] ^ (ROTL32(key[i], 1)) ^ (ROTL32(key[i], 8));
      round_keys[ + KEY_SIZE_128_128] = key[i] ^ (ROTL32(key[i], 1)) ^ (ROTL32(key[i], 11));
  }
```

The key scheduling equation for `rk[(i+k/w)^1]` has been fixed to use `(K[i] <<< 11)` instead of `(K[i] <<< 16)`.

ChatGPT와 암호 알고리즘 구현: CHAM

- 구현한 코드를 가동해본 결과, 테스트벡터와 일치함
  - 하지만 구현 중간에 너무 많은 문제가 발생



```
CHAM-64/128
Secret Key: 0100 0302 0504 0706 0908 0b0a 0d0c 0f0e
Plaintext: 1100 3322 5544 7766
Round Keys: 0301 0705 0b09 0f0d 1311 1715 1b19 1f1d 151e 0308 3932 2f24 4
Ciphertext: 6579 1204 123f e5a9

CHAM-128/128
Secret Key: 03020100 07060504 0b0a0908 0f0e0d0c
Plaintext: 33221100 77665544 bbaa9988 ffeeddcc
Round Keys: 07070303 0f0f0b0b 17171313 1f1f1b1b 39222f34 150e0318 617a776
Ciphertext: d05419ee 9f118f4c 99e36469 1c885ec1

CHAM-128/256
Secret Key: 03020100 07060504 0b0a0908 0f0e0d0c f3f2f1f0 f7f6f5f4 fbfaf9f
Plaintext: 33221100 77665544 bbaa9988 ffeeddcc
Round Keys: 07070303 0f0f0b0b 17171313 1f1f1b1b e6e6e2e2 eeeeeaea f6f6f2f
Ciphertext: 027377dc 120b5651 8f839b95 5e5ec075
```

CHAM-64/128

```
secret Key : 0x0100 0x0302 0x0504 0x0706 0x0908 0x0b0a 0x0d0c 0x0f0e
plaintext  : 0x1100 0x3322 0x5544 0x7766
ciphertext : 0x6579 0x1204 0x123f 0xe5a9
```

CHAM-128/128

```
secret Key : 0x03020100 0x07060504 0x0b0a0908 0x0f0e0d0c
plaintext  : 0x33221100 0x77665544 0xbbaa9988 0xffeeddcc
ciphertext : 0xd05419ee 0x9f118f4c 0x99e36469 0x1c885ec1
```

CHAM-128/256

```
secret Key : 0x03020100 0x07060504 0x0b0a0908 0x0f0e0d0c
             0xf3f2f1f0 0xf7f6f5f4 0xfbfaf9f8 0xfffefdfc
plaintext  : 0x33221100 0x77665544 0xbbaa9988 0xffeeddcc
ciphertext : 0x027377dc 0x120b5651 0x8f839b95 0x5e5ec075
```

결론

- ChatGPT를 사용한 프로그래밍의 시도는 많음
  - **단순한 수준의 프로그래밍부터 복잡한 수준까지 모두 가능**
- 복잡한 프로그래밍의 경우, 전문가가 아니면 GPT를 신뢰하기 어려움
  - 이전에 학습한 것을 잊기도 하고 같은 실수를 반복하기도 함
- **암호 알고리즘의 구현도 가능함**
  - 널리 알려진 알고리즘은 손쉽게 구현 가능
  - 잘 알려지지 않은 알고리즘은 구현이 까다로움
  - 많은 시행착오를 거치면 구현이 가능
  - 하지만 암호 알고리즘 내부를 잘 모른다면
    GPT가 잘못 구현해도 발견하지 못할 수 있음