Q1. What is the difference between a code and a cipher?

Ans1: The terms "code" and "cipher" are often used interchangeably in casual conversation, but in the context of cryptography, they have distinct meanings and are used for different types of secrecy.

Code

A code is a system of symbols, letters, or words used to represent others, typically for the purpose of secrecy. In traditional coding systems, complex ideas, phrases, or entire sentences are replaced with a word, number, or symbol. Codes work at the level of meaning; that is, a single code word could represent a phrase, a place, a person, or an action. Historically, codes were used to condense lengthy messages or to keep specific information secret.

For example, in a military code, the word "Eagle" might represent a particular military unit, or the phrase "The eagle has landed" could mean a specific operation has been completed successfully.

Cipher

A cipher, on the other hand, is a method for transforming individual letters or small groups of letters in a message to obscure the content. Ciphers work at the level of the individual letters or small groups of letters (like pairs or triples), rather than whole words or phrases as codes do. The process of encoding a message using a cipher is known as encryption, and the process of turning it back into readable text is known as decryption. Ciphers are categorized into two main types: substitution ciphers, where each letter in the plaintext is replaced with another letter or symbol, and transposition ciphers, where the positions of the letters are shifted according to a certain system.

For example, in a simple substitution cipher like the Caesar cipher, each letter in the plaintext might be shifted three places down the alphabet. So, the word "HELLO" encrypted with a Caesar cipher with a shift of 3 would become "KHOOR".

Key Differences

Level of Application: Codes apply to words, phrases, or sentences, replacing them with symbols, numbers, or other words. Ciphers apply to the individual letters or small groups of letters, altering their arrangement or replacing them with other letters based on a systematic method.
Purpose: Codes are often used to condense messages or to replace specific, predetermined phrases or terms with symbols or other words. Ciphers are designed to transform a message in such a way that it becomes unreadable to anyone who does not have the key to decrypt it.
Key Usage: Ciphers usually require a key for encryption and decryption, making them more flexible and suitable for a wide range of secrecy needs. Codes, especially historical ones, might

not use a key in the same way, and once a codebook is compromised, the secrecy of the coded messages is also compromised.
In modern cryptography, ciphers are far more common than codes, especially with the advent of computerized encryption algorithms that can quickly and securely encrypt large amounts of data.

Q2. In the course notes, we demonstrated an Auto-Key Vigenère cipher using the plaintext following the keyphrase. Repeat the example encryption in the notes by following the keyphrase with the ciphertext.

Ans2: **Output:**
Ciphertext: JUKVKLIZTYQGNNPFUYVRDNMIFUZMENFBABBSUHFUZCYIZKQMZ

```
function encryptAutoKeyVigenere(plaintext, keyphrase) {
    const ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    let key = keyphrase;
    let ciphertext = "";

    for (let i = 0; i < plaintext.length; i++) {
        const plaintextChar = plaintext[i];
        const keyChar = key[i % key.length];
        const plaintextIndex = ALPHABET.indexOf(plaintextChar);
        const keyIndex = ALPHABET.indexOf(keyChar);

        // Calculate the index for ciphertext character and get the character
        const cipherIndex = (plaintextIndex + keyIndex) % ALPHABET.length;
        const cipherChar = ALPHABET[cipherIndex];

        // Append the cipher character to the ciphertext and the key
        ciphertext += cipherChar;
        key += cipherChar; // This is the unique step for auto-key with ciphertext extension
    }

    return ciphertext;
}

const plaintext = "TAKEACOPYOFYOURPOLICYTONORMAWILCOXONTHETHIRDFLOOR";
const keyphrase = "QUARK";
const ciphertext = encryptAutoKeyVigenere(plaintext, keyphrase);
```

```
console.log("Ciphertext:", ciphertext);
```

Q3. Write a Node.js command line application that takes in four command line arguments. The first is either `-e` or `-d` (for encrypt and decrypt, respectively). The second is either a message to encrypt or a ciphertext to decrypt. The third is the key. The fourth is an initialization vector. If encrypting, the program assumes the string argument is a utf-8 encoded string and outputs the AES256-CBC encoding of the input, as a hexadecimal string. If decrypting, the string argument is assumed to be a hexadecimal representation of a byte sequence, and the output should be the decrypted string.

Ans3:

```
// aes256cbc.js
const crypto = require('crypto');

function encrypt(text, key, iv) {
    const cipher = crypto.createCipheriv('aes-256-cbc', Buffer.from(key),
iv);
    let encrypted = cipher.update(text, 'utf8', 'hex');
    encrypted += cipher.final('hex');
    return encrypted;
}

function decrypt(encryptedText, key, iv) {
    const decipher = crypto.createDecipheriv('aes-256-cbc',
Buffer.from(key), iv);
    let decrypted = decipher.update(encryptedText, 'hex', 'utf8');
    decrypted += decipher.final('utf8');
    return decrypted;
}

function main() {
    const args = process.argv.slice(2);
    if (args.length !== 4) {
        console.log('Usage: node aes256cbc.js -e|-d "message|hex" key iv');
        process.exit(1);
    }

    const [action, input, key, iv] = args;
    if (key.length !== 32) {
        console.error('The key must be exactly 32 bytes long.');
        process.exit(1);
    }

    if (iv.length !== 16) {
        console.error('The IV must be exactly 16 bytes long.');
        process.exit(1);
    }

    if (action === '-e') {
        console.log(encrypt(input, key, iv));
    } else if (action === '-d') {
        console.log(decrypt(input, key, iv));
    } else {
        console.log('Invalid action. Use -e for encryption or -d for
decryption.');
        process.exit(1);
    }
}

main();
```

Q4. Simulate RSA-512 encryption and decryption (WITH THE USUAL DISCLAIMERS THAT THIS SIZE IS TOO SMALL AND IS ONLY USED FOR EDUCATIONAL PURPOSES AND DO NOT DO THIS STUFF YOURSELF) where p=100392089237316158323570985008687907853269981005640569039457584007913129640081 and q=90392089237316158323570985008687907853269981005640569039457584007913129640041 and e=65537. Use a 60-byte block size.

a. What is N?
b. What is d?
c. Given the message "Scaramouche, Scaramouche, will you do the Fandango? 💃🏾", what is the resulting ciphertext? Show your answer as a byte sequence written in hex.
d. Decrypt the ciphertext. What did you get back? Yes, you should get back the original message. But be honest. Show your work, as they say.

Ans: **N (the modulus):**
9074650689060089248199307400991055468098862103321403389047443270291029585149
4374128997882303074774615183542918015346609049404244318109659484119314160833
21

9074650689060089248199307400991055468098862103321403389047443270291029585149
4374128997882303074774615183542918015346609049404244318109659484119314160833
21

d (the private exponent):
3440604854078842449902442746842634638010902628184540510109569714515334896803
1566936305781510120413168154946928973840296197553039132498283075405102604062
73

3440604854078842449902442746842634638010902628184540510109569714515334896803
1566936305781510120413168154946928973840296197553039132498283075405102604062
73

Given the message "Scaramouche, Scaramouche, will you do the Fandango? 💃🏽", the resulting ciphertext as a byte sequence written in hex is:

`1510726ec4756e595c4b5ce1f3a1974798a34369eb8f43f7462d4093f30973994849a
5b63d6b28e33c2200bfea7f7005bd7642e74302832b739be60d966a926b`

Decryption of this ciphertext yielded back the original message:

"Scaramouche, Scaramouche, will you do the Fandango? 💃🏽"

Q5. What is the sha384 digest of the phrase "Російський військовий корабель, іди нахуй"?

Ans:
`c358ff602ada470dfb85fad41bd1fe277d587ace98d09c7eb70f48ef1048b76a2ec1103f6
7d54871cd18046cbd6fe816`