PART B: Cryptography

1) One-way functions are a fundamental element of modern cryptography, allowing for the creation of secure encryption and decryption systems. These functions are designed to be easy to compute in one direction, but difficult to reverse or invert. This makes them useful for protecting sensitive data and preventing unauthorized access.

   In RSA (Rivest-Shamir-Adleman) cryptography, one-way functions are used to create a secure encryption and decryption system. RSA relies on the modular exponentiation function, which involves raising a base number to a specific power modulo a prime number. This function is difficult to reverse or invert, making it suitable for use in a one-way function.

   ECC (Elliptic Curve Cryptography) also uses one-way functions to create a secure encryption and decryption system. In ECC, the one-way function is the elliptic curve discrete logarithm function, which involves finding the discrete logarithm of a point on an elliptic curve to a base point. This function is also difficult to reverse or invert, making it suitable for use in a one-way function.

   Overall, one-way functions play a vital role in the security of both RSA and ECC cryptographic systems. They make it difficult for an attacker to derive the original input from the output, helping to ensure the confidentiality and integrity of sensitive data being transmitted or stored. Proper implementation of one-way functions is crucial for the security of modern cryptographic systems.

2) In ECC (Elliptic Curve Cryptography), the private key is a unique, confidential value that is used in conjunction with a one-way function to generate the public key. The private key is typically represented by a letter "d" and is kept secret by the owner. It is used to create the public key, which is represented by a letter "Q" and is made publicly available.

   The process of generating the public key from the private key involves the use of the elliptic curve point multiplication function. This function involves multiplying a point on an elliptic curve (called the base point) by the private key, modulo a prime number (called the modulus). The result of this operation is a new point on the curve, which is called the public key. It is important to note that the public key is not the same as the private key, but is derived from it through the use of the one-way function.

   The public key in ECC is used for a variety of purposes, including the encryption of messages and the verification of digital signatures. Anyone can use the public key to send encrypted messages to the owner of the private key, or to verify the owner's digital signatures. However, only the owner of the private key is able to decrypt the message or create a valid digital signature, as the private key is required for these operations.

   Overall, the private and public keys in ECC play a crucial role in creating a secure encryption and decryption system. The private key is kept secret and is used to generate the public key, which is made publicly available and used for a variety of security-related purposes. The use of the one-way function ensures that the private key remains confidential, while the public key can be used for the secure transmission of information and the authentication of digital signatures.

3) Both RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) are widely used cryptographic algorithms that are used to protect the confidentiality and integrity of data transmitted over networks, as well as authenticate digital signatures. However, there are some key differences between the two algorithms in terms of key size and encryption strength.

   One of the main differences between RSA and ECC is the amount of key size required to achieve the same level of security. In general, ECC requires a smaller key size to provide the same level of security as RSA. For instance, a 256-bit ECC key is considered to be equivalent in security to a 3072-bit RSA key. This is because ECC uses a different type of mathematical problem, known as the elliptic curve discrete logarithm problem, to create its one-way functions, which allows it to provide the same level of security with a smaller key size.

   Another difference between RSA and ECC is the level of encryption strength they provide. RSA relies on the difficulty of factoring large composite numbers to provide secure encryption. However, as computational power increases, RSA may become vulnerable to attacks that can factor larger numbers in a reasonable amount of time. On the other hand, ECC is based on the elliptic curve discrete logarithm problem, which is believed to be resistant to attacks even with increasing computational power. This makes ECC a potentially stronger choice for long-term encryption.

   In conclusion, both RSA and ECC offer strong encryption and authentication capabilities, but there are some key differences between the two algorithms in terms of key size and encryption strength. ECC generally requires a smaller key size to provide the same level of security as RSA, and may also provide stronger encryption in the long run due to its resistance to attacks with increasing computational power.

4) To encrypt the message M=59 using the RSA public key (N=713, e=7), we can follow these steps:

   ⇨ RSA utilizes a one-way function known as the modular exponentiation function for the purpose of encrypting and decrypting messages. This function involves raising a number (referred to as the base) to a specified power (referred to as the exponent) modulo a prime number (referred to as the modulus). In this case, the base is the message M, the exponent is the public key e, and the modulus is the public key N.

   ⇨ In order to encrypt the message M using the RSA public key, we first need to calculate the ciphertext C using the formula C = M^e mod N. This formula represents the modular exponentiation function, with M being the base, e being the exponent, and N being the modulus.

   ⇨ Substituting the values for M, e, and N into the formula gives us C = 59^7 mod 713. This equation can be used to calculate the ciphertext C using the RSA public key.

- ⇨ We can then use a calculator or computer program to find the value of C. In this case, the result is C = 356.

- ⇨ The ciphertext C is the encrypted version of the message M. In this instance, the ciphertext C is 356. It is important to note that the RSA public key (N, e) is made publicly available and can be used by anyone to send encrypted messages to the owner of the private key. The private key is kept secret and is used to decrypt the ciphertext. To decrypt the ciphertext C and recover the original message M, the recipient of the message would need to use the private key, which is kept secret by the sender. The private key is used to perform the inverse operation of the encryption process and recover the plaintext that was being encrypted

5) Eve claims that she can use the following program to find the prime numbers p and q that were used by RSA to calculate the public key N and thus decrypt the message sent by Alice.

```
i = 2
N = 713
while i < N/2:
if N % i == 0:
print("p=", i, "q=", N/i)
break
```

Eve claims that by running a certain program, she is able to determine the prime numbers p and q that were used to calculate N in Bob's RSA public key (N=713). She claims that she can then use this information to decrypt the encrypted message sent by Alice (which is 16). To verify her claim, we can test the program with the given input values and see the output. The input values are N=713 and i=2.

The program works by checking if N is divisible by i. If it is, it prints the values of p and q and breaks the loop. We can follow the program's execution step-by-step:

- ➔ Set the value of i to 2 and the value of N to 713.
- ➔ Enter the while loop, which will continue to execute as long as i is less than N/2.
- ➔ The program checks if the N is divisible by i. In our case, 713 is not divisible by 2, so the program skips the print statement and continues to the next iteration of the loop.
- ➔ The program increments the value of i by 1.
- ➔ The program checks if the N is divisible by i. In our case, 713 is divisible by 3, so the program prints "p=3 q=237" and then it breaks the loop.
- ➔ The program then finishes the execution.

The output of the program is "p=3 q=237". Eve was able to find the prime numbers p and q that were used to calculate N by running the program and checking if N was divisible by each integer between 2 and N/2. However, this does not allow her to decrypt the message sent by Alice without having Alice's private key. In order to decrypt the message, Eve would need to know the private key, which is kept secret and is used to perform the inverse operation of the encryption process. Without the private key, Eve is unable to decrypt the message and recover the original message.

6) Eve could potentially use the values of p and q that she obtained in part (5) to find the decrypted message, but only if the RSA algorithm is not strong enough or if Bob has made a mistake in implementing it.

   If the RSA algorithm is not strong enough, it may be possible for Eve to use the values of p and q to derive the private key, which she could then use to decrypt the message. This would require Eve to have access to a sufficiently powerful computer or to have developed a successful attack against the RSA algorithm.

   Alternatively, if Bob has made a mistake in implementing the RSA algorithm, it may be possible for Eve to exploit this mistake and use the values of p and q to derive the private key and decrypt the message. For example, if Bob has not properly protected his private key or has made a mistake in generating the public key, Eve may be able to exploit this vulnerability and decrypt the message.

   Overall, it is important for Bob to ensure that the RSA algorithm is strong enough and that he has implemented it correctly in order to protect the confidentiality of the message and prevent Eve from being able to decrypt it. So, it is very important to use a strong encryption algorithm and to implement it correctly in order to protect the confidentiality of the message.

7) Here is a simple pseudocode program to show how one could try to find the private key in ECC:

   - ⇨ Set the starting point on the curve to P and the public key to Q.
   - ⇨ Set a variable "i" equal to 1.
   - ⇨ Calculate the point R by performing the operation "P . i" and assign it to R.
   - ⇨ Check if R is equal to Q. If it is, print "Private key found: i" and exit the program.
   - ⇨ If R is not equal to Q, increment the value of i by 1 and go back to step 3.

```
def find_private_key_in_ECC(P, Q):
i = 1
while Q != P:
# increment i by 1
i += 1
# perform point multiplication on P using i as the private key
P = P . i
return i
```

   This program works by starting at point P and repeatedly performing the "." operation with increasing values of i until it arrives at the point Q, which is the public key. If the program finds a value of i that results in R being equal to Q, it prints the value of i as the private key.

This program can find the private key in ECC if the "." operation is defined and if the private key is within the range of values that are checked by the program. The program will only find the private key if it is able to perform the "." operation with the correct value of i and arrive at the point Q, which is the public key. If the private key is not within the range of values checked by the program, or if the "." operation is not defined correctly, the program will not be able to find the private key.

8) Here is the pseudocode:

```
function calculate_public_key(k, P):
Q = P
for i in range(1, k):
# Set Q to Q + P
Q = Q + P
return Q
```

To explain the circumstances under which a program can find the private key in ECC, it is important to understand the concept of a one-way function. In ECC, the private key is derived from the public key through the use of a one-way function called the elliptic curve point multiplication function. This function involves multiplying a point on an elliptic curve (called the base point) by the private key modulo a prime number (called the modulus). The result of this multiplication is a new point on the curve, which is the public key.

Since the elliptic curve point multiplication function is a one-way function, it is easy to calculate the public key given the private key, but difficult to calculate the private key given the public key. This means that it is relatively easy to encrypt a message using the public key, but very difficult to decrypt the message without the private key.

Therefore, in order for a program to find the private key in ECC, it would need to be able to efficiently invert the elliptic curve point multiplication function, which is currently considered to be computationally infeasible. This means that it is highly unlikely that a program could be developed that could find the private key in ECC, even with access to the public key and starting point on the curve.