PART B: Cryptography

1) A one-way function is a type of mathematical function that is easy to compute in one direction, but difficult to reverse or invert. In cryptography, one-way functions are used to create secure cryptographic systems by making it difficult for an attacker to derive the original input from the output.

   In RSA (Rivest-Shamir-Adleman) cryptography, one-way functions are used to create a secure encryption and decryption system. RSA uses a one-way function called the modular exponentiation function, which involves taking a number (called the base) and raising it to a specific power (called the exponent) modulo a prime number (called the modulus). The output of this function is difficult to reverse or invert, making it a good choice for use in a one-way function.

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

   Overall, one-way functions play a critical role in the security of both RSA and ECC cryptographic systems by making it difficult for an attacker to derive the original input from the output. This helps to ensure the confidentiality and integrity of sensitive data being transmitted or stored.

2) In ECC (Elliptic Curve Cryptography), the private key is a secret value that is used to create the public key. The public key is then used to encrypt messages or verify digital signatures.

   To generate the private key in ECC, a random integer is chosen and kept secret. This integer is called the private key and is usually represented by a letter "d". The private key is used to create 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 called the public key.

   In ECC, the public key is usually represented by a letter "Q". 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 is made publicly available and can be used by anyone to send encrypted messages to the owner of the private key or to verify the owner's digital signatures.

   Overall, the private and public keys in ECC are used to create a secure encryption and decryption system that allows for the confidential transmission of messages and the authentication of digital signatures. The private key is kept secret and is used to create the public key, which is made publicly available and used to encrypt messages or verify signatures.

3) RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) are both commonly used cryptographic algorithms that are used to secure the transmission of data and to 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 size of the key required to achieve the same level of security. In general, ECC requires a smaller key size to achieve the same level of security as RSA. For example, 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 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 encryption strength provided by each algorithm. RSA is based on the difficulty of factoring large composite numbers, which makes it relatively secure against attacks using currently available computational resources. However, as computational power increases, RSA may become vulnerable to attacks that can factor larger numbers in a reasonable amount of time. ECC, on the other hand, is based on the difficulty of the elliptic curve discrete logarithm problem, which is thought to be resistant to attacks even with increasing computational power. This makes ECC a potentially stronger choice for encryption in the long term.

   Overall, RSA and ECC both provide 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 achieve the same level of security as RSA, and may also provide stronger encryption in the long term 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 use the following steps:

   ⇨ RSA uses a one-way function called the modular exponentiation function to encrypt and decrypt messages. The modular exponentiation function involves raising a number (called the base) to a specific power (called the exponent) modulo a prime number (called 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.

   ⇨ 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.

- ⇨ Substitute the values for M, e, and N into the formula: C = 59^7 mod 713. This gives us the equation for calculating the ciphertext C using the RSA public key.

- ⇨ Calculate the value of C using a calculator or computer program. In this case, the result is C = 356.

- ⇨ The ciphertext C is the encrypted version of the message M. In this case, 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 original message M.

5) Using the given program, Eve can find the prime numbers p and q that were used to calculate the RSA public key N. However, this does not allow her to decrypt the message sent by Alice without having Alice's private key.

To verify Eve's claim, we can run the program with the given input values and see what the output is. The input values are N=713 and i=2. The program checks if N is divisible by i, and if it is, it prints the values of p and q and breaks the loop.

Step-by-step, the program works as follows:

- ⇨ 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.

- ⇨ Check if N is divisible by i. In this case, 713 is not divisible by 2, so the program skips the print statement and continues to the next iteration of the loop.

- ⇨ Increment the value of i by 1.

- ⇨ Check if N is divisible by i. In this case, 713 is divisible by 3, so the program prints "p=3 q=237" and breaks the loop.

- ⇨ The program finishes executing.

- ⇨ 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) If Eve were able to find the prime numbers p and q that were used to calculate the RSA public key N, she could potentially use this information to decrypt the message sent by Alice. However, this would require additional knowledge and steps beyond simply knowing p and q.

   In RSA cryptography, the private key is created using the prime numbers p and q, as well as a few other values. Specifically, the private key consists of the prime numbers p and q, as well as the private exponent d and the modulus N. The private exponent d is calculated using the extended Euclidean algorithm, and is used to perform the inverse operation of the modular exponentiation function used in the encryption process.

   To decrypt the message using p and q, Eve would need to know the value of d and the modulus N. Without these values, she would not be able to use p and q to decrypt the message.

   In general, RSA is considered to be a strong cryptographic algorithm, as it is based on the difficulty of factoring large composite numbers. However, if the prime numbers p and q are not chosen properly or are not kept secret, it is possible for an attacker to derive the private key and decrypt the message. In this case, it is possible that Bob may have missed something in the process of generating and protecting the private key, such as failing to properly secure the prime numbers p and q. This could allow Eve to find the decrypted message using p and q. However, it is also possible that the RSA algorithm itself is not strong enough to protect against certain types of attacks, such as those that are able to factor large composite numbers in a reasonable amount of time. In this case, the weakness would not be due to a mistake made by Bob, but rather to the inherent limitations of the RSA algorithm.

7) Here is a pseudocode example of how one could try to find the private key in ECC (Elliptic Curve Cryptography) using two points P and Q on the curve:

```
function find_private_key(P, Q):
    i = 1
    while i < infinity:
        R = i * P
        if R == Q:
            return i
    i = i + 1
return "private key not found"
```

In this example, the function takes two points P and Q as input and tries to find the private key by iterating through values of i and calculating the point R as the result of multiplying P by i. If R is equal to Q (the public key), the function returns the value of i as the private key. If the function reaches the end of the loop and has not found the private key, it returns the message "private key not found".

This program can find the private key in ECC if the starting point P and the public key Q are known, and if the operation "." (or "+") for adding two points on the curve has been defined. In these circumstances, the program can iterate through values of i and check if the result of multiplying P by i is equal to Q. If it is, the value of i is the private key.

It is important to note that this program is a simplified example and may not be sufficient to find the private key in all cases. In practice, finding the private key in ECC can be more complex and may require more advanced techniques. Additionally, the security of ECC relies on the difficulty of the elliptic curve discrete logarithm problem, which makes it computationally infeasible to find the private key from the public key. This makes ECC a secure method for creating a key exchange and protecting data, as it is believed to be practically impossible for an attacker to find the private key from the public key for large keys. However, it is important to note that the security of ECC may be affected by certain attacks, such as the use of a weak curve or the use of a faulty implementation of the algorithm. It is important for users of ECC to carefully consider these factors when using the algorithm for security purposes.

8) Here is a simple pseudocode example of a program that can efficiently calculate the public key in ECC given a private key k and a starting point P on the curve:

```
function calculate_public_key(k, P):
  Q = P
  for i in range(1, k):
    Q = Q . P
  return Q
```

This function takes a private key k and a starting point P as input and calculates the public key Q by repeatedly adding P to itself k-1 times. This method is efficient because it only requires a single point multiplication for each iteration of the loop, which is much faster than other methods that may require multiple point multiplications.

It is important to note that this is a simplified example and may not be sufficient for all situations. In practice, calculating the public key in ECC may require more advanced techniques and may depend on the specific implementation of the algorithm being used.