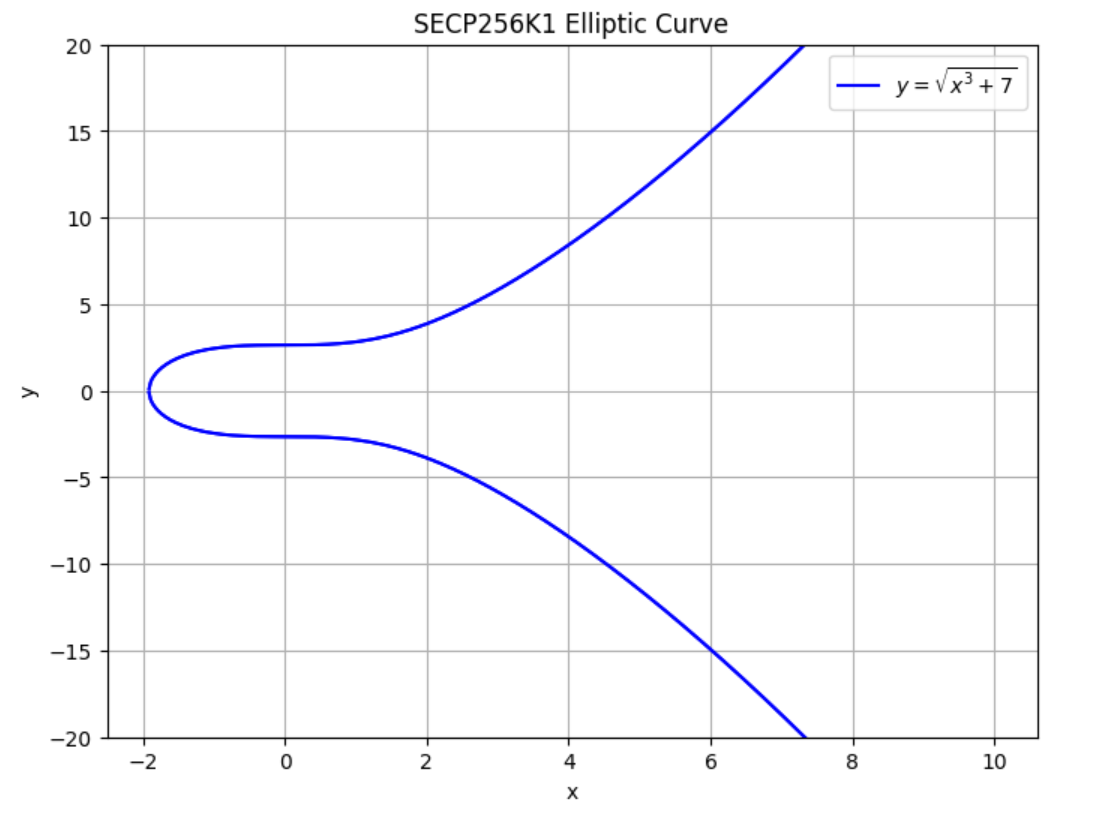
**1. Explanation and Visualization of the SECP256K1 Elliptic Curve**

Elliptic curve cryptography (ECC) utilizes mathematical curves with unique attributes for the purpose of maintaining security when dealing with cryptographic keys. The distinctive mathematical SECP256K1 elliptic curve is highly prevalent within Bitcoin's blockchain network. Being classified by the equation over a finite field, that precise equation guarantees the consistent and undeterred shape of the curve, which stands crucial in implementing complex mathematical algorithms required in cryptographic systems.

**Group Law on Elliptic Curves**

1. Point Addition: The group law is a set of rules that let you add two points on an elliptic curve in such a way that you magically get a third point. For points P, Q on the curve, draw a straight line through both points. That line will cross the curve at exactly one other point. You reflect this third intersection point over the x-axis to get the sum R=P+Q. This operation is commutative (P+Q = Q+P) and associative, which means you can add more than two point in any order and get the same result.
2. Scalar Multiplication: This operation takes a point P on the curve and multiplies it by a number n, which is just adding P to itself n times. Scalar multiplication is always depicted as an iteration of the point addition law. It is one of the absolutely key features of elliptic curve cryptography, especially for going from private to public keys.

**Visualization of the SECP256K1 Elliptic Curve**  
The curve corresponding to SECP256K1 is given by . This curve is drawn below. What's interesting about this curve’s shape is the symmetry around the x-axis. This symmetry is at the heart of the mathematical properties that allow for the cryptographic operations defined.



The graph depicted above represents the elliptic curve SECP256K1, denoted by the equation . The curve is symmetric with respect to the x-axis, and this plot is useful to visualize the geometric operations applied to the curve, like point addition and scalar multiplication.

**2.Demonstrating Geometric Addition and Scalar Multiplication on the SECP256K1 Curve**

Point addition and scalar multiplication are the two fundamental operations performed on the points of an elliptic curve, which are essential for Elliptic Curve Cryptography (ECC) to work, including key generation and encryption/decryption algorithms.

**Geometric Addition of Points**

Addition of Points involves taking two points on the elliptic curve and finding their sum according to the group law defined for the curve. It works roughly like this:

1. Given Points: Take two points, P and Q, on the curve.
2. Draw a Line: Draw a straight line that intersects both P and Q. Due to the curvature of the curve, this line will intersect the curve at only one more point.
3. Find the Third Intersection: Identify the third point of intersection, T.
4. Reflect Across the X-Axis: The sum R=P+Q is shown by reflecting point T across the x-axis.

The operation of point addition satisfies commutativity (P+Q=Q+P) and associativity ((P+Q) +R=P+(Q+R)). Endpoint addition is thus a fundamental building block for elliptic curve cryptosystems.

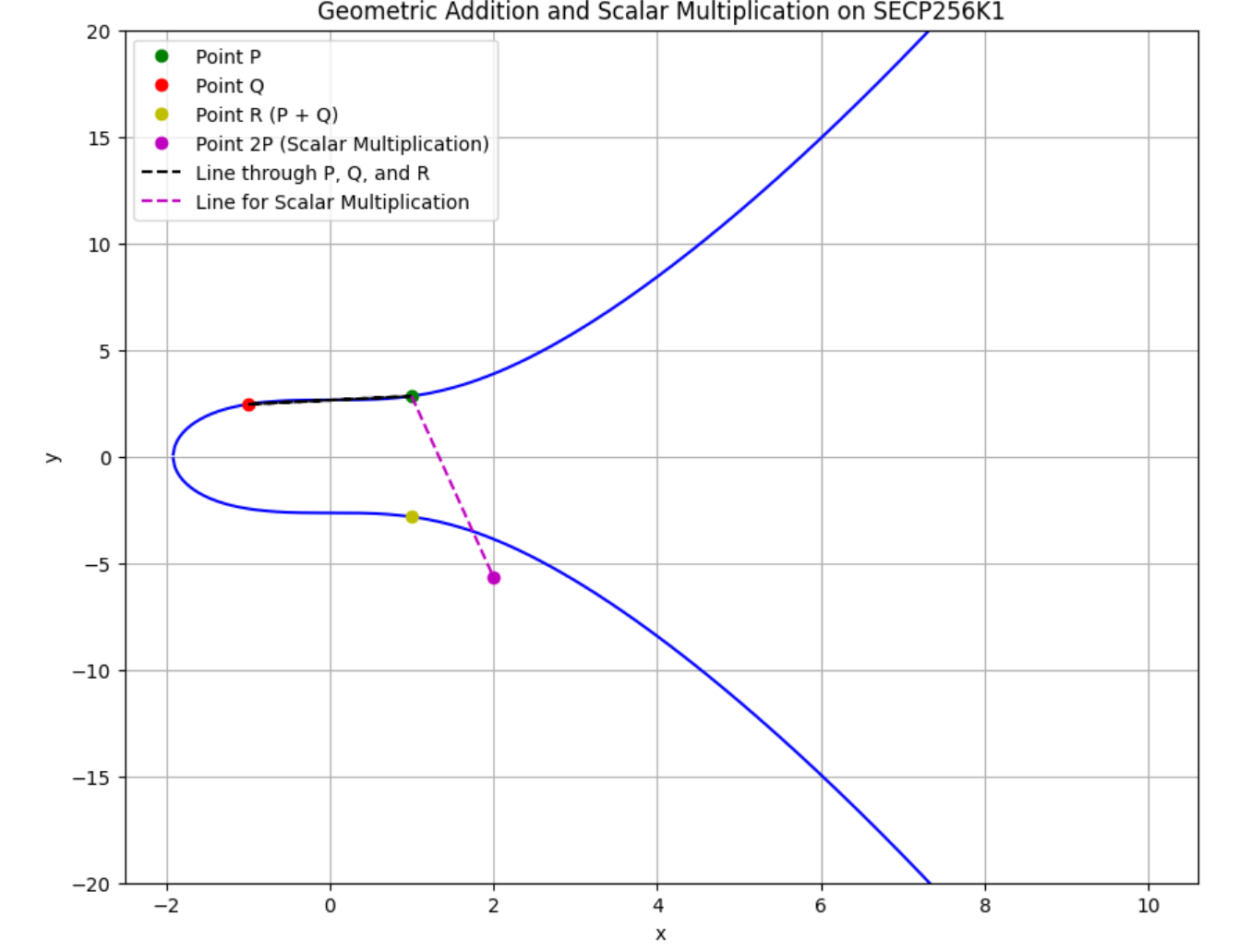
**Scalar Multiplication of Points**

Scalar Multiplication is the procedure of point P multiplication on the curve by the scalar (integer) n. Which is actually the adding of P to itself n times. This procedure can be made faster by using the "double and add" method, which is crucial for fast ECC calculations.

1. Given Point: chose a point P on the curve.
2. Scalar Value: chose a scalar n.
3. Multiply: perform operation n⋅P by adding P to itself n times.

Scalar multiplication is most often used for the generation of public keys from a private key in ECC, it is vital for encryption algorithms, and digital signature algorithms.

**Visual Demonstration**



* Geometric Addition: The points P (green) and Q (red) are added together by drawing a line that intersects the curve at a third point. That third point is then reflected across the x -axis to find the result point R (yellow) which represents P + Q
* Scalar Multiplication: I've also shown an approximate scalar multiplication here, where a point P is added to itself. In this case it's 2P (magenta), but in general 2P would mean "add P to itself exactly once". The magenta dashed line isn't quite right to use for geometric calculations, but helps show the conceptual idea.

**3. Finite Prime Fields in Elliptic Curve Cryptography (ECC)**

Elliptic Curve Cryptography (ECC) operates on elliptic curves over finite fields, ensuring security and efficiency in cryptographic processes. A finite field , also known as a Galois field, is a set of finite elements on which addition, subtraction, multiplication, and division (excluding division by zero) operations are defined and behave similarly to those operations on integers.

For ECC, the most commonly used finite fields are prime fields and binary fields . A prime field consists of the integers where is a prime number, and the operations of addition, subtraction, and multiplication are performed modulo .

**How Elliptic Curves Are Restricted to a Finite Prime Field**

1. **Definition Over :** An elliptic curve over is defined by an equation of the form , where and are coefficients that satisfy the condition that (to ensure the curve is non-singular), and all arithmetic is performed modulo .
2. **Finite Set of Points:** Since we're working modulo , both and take values within the finite field , resulting in a finite set of points on the curve. This property is crucial for cryptographic applications, as it allows the definition of discrete logarithm problems, which are hard to solve and form the basis of ECC's security.
3. **Modular Arithmetic:** In order to ensure the points are contained in the field, and maintain the closed and finite nature of the points in the curve set, modular arithmetic is carried out on them.
4. **Cyclic Subgroups**: Within the elliptic curve sets of points, a subset of them produce cyclic subgroups through a generator point G. These cyclic and the complexity of the discrete logarithm problem in these groups are what are at the center of key generation, encryption and signature schemes of ECC.

**4. Subgroup Order**

The subgroup order, in the context of Elliptic Curve Cryptography (ECC), is the count of points, including the “point at infinity”, which is the “identity element”, within a subgroup determined by an individual point on the elliptical curvature.

1. **Definition:** Essentially, if you start with a point called G on an elliptic curve and you keep adding G to itself (scalar multiply), you’re eventually going to hit the point at infinity, having effectively gotten back to where you started. The number of unique points you pass along the way, including the initial point G and the point at infinity, is the subgroup order.
2. **Cyclic Subgroups:** Among the elliptic curves in this class, cyclic subgroups arise. By repeatedly adding G to itself, a subgroup can be created that is defined by a generator point G. In ECC, the subgroup order is critical because it is the value that measures cryptographic algorithm strength. To illustrate, the Elliptic Curve Discrete Logarithm Problem (ECDLP), over which ECC security is rooted, is more complex as the subgroup order increases.
3. **Choosing Subgroup Order:** An important concept when it comes to applying cryptography is the choice of an elliptic curve which has subgroups of prime order. The purpose of this is to make it hard, and by hard we mean very hard, for the adversary to compute/find a scalar (times the point add itself) if he has access only to the initial and final points on the curve, especially when the order of the subgroup is a big prime number.

In summary, the subgroup order is a fundamental concept in ECC, which directly affects the security and efficiency of cryptographic system. The selection of curves with the subgroup order of a large prime order is the important issue in constructing secure cryptographic system.

**5. Base Point**

With Elliptic Curve Cryptography (ECC), we generate a public/private key pair from a point on the curve known as the base point (or sometimes the generating point).

1. **Role in ECC:** The Base Point is used as a reference for calculating scalar multiplication which entails repeatedly adding the point to itself. The outcome of doing so is that a fresh point on the curve is revealed which may, due to the mathematical properties of elliptic curves, be unpredictably far distant. Consequently, cryptographic power is provided.
2. **Selection Criteria:** Selection Criteria: The Base Points of optimal elliptic curves are not chosen at random. They are selected to have certain properties to enhance the security of the cryptographic system, such as:
   * The base point must be on the curve.
   * The subgroup generated by the base point should have a large prime order, which enhances the difficulty of resolving the distinct logarithm problem on the curve.
   * The coordinates of the base point are usually part of the curve's specification and are chosen to facilitate efficient computation.
3. **Impact on Cryptography:** The choice of the base point, along with the curve's parameters, determines the security and performance of ECC systems. A well-chosen base point ensures that the resulting cryptographic operations are both secure against attacks and efficient for practical use.

**6. Geometric Addition and Scalar Multiplication Over the Prime Field**Performing operations over a prime field involves applying modular arithmetic to the coordinates of the points on the elliptic curve.

Considering a simplified elliptic curve defined over a prime field with the equation . Demonstrating:

Geometric Addition of two points and on .  
Point Scalar Multiplication by a scalar on .

Picking two points and on the curve (within our finite field) and calculate their sum . Performing scalar Point Scalar Multiplication by a scalar .

Taking modular operations into account:

* For addition , Calculating the slope , then and .
* For scalar multiplication (nP), repeatedly adding to itself times, using the point addition formula each time.

**Geometric Addition of Points 𝑃 and 𝑄 Over 𝐹19**  
Given points and , we aim to find .

**Calculating the Slope (𝜆):**

* The slope is determined by .
* Calculation yields .
* Since 11 has a multiplicative inverse of 7 in .

**Finding 𝑅=𝑅, 𝑦𝑅**

* Using , we compute , which simplifies to .
* Next, , resulting in .
* Thus, is the point resulting from .

**Scalar Multiplication of Point by Scalar 2 Over**   
Given , we find .

Calculating the Slope for Doubling :

* The slope for doubling, , is computed by , with .
* This gives .
* With the multiplicative inverse of 2 being 10 in , we find .

**Finding :**

* The -coordinate is , simplifying to .
* The y-coordinate is , resulting in .
* Hence, represents the point after scalar multiplication by 2.

**7. ECC Keys Generation**  
The key generation process in Elliptic Curve Cryptography (ECC) involves generating keys using the principles of discrete logarithm problems on elliptic curves over finite fields. Scalar multiplication is used in this process. Scalar multiplication is straightforward in one direction, but it is nearly impossible to reverse it computationally, which is why ECC’s security is built on this mathematical problem. Let's look at the key generation process in more detail;

1. Choose an elliptical curvature E over a finite field and a base point G on that curve. The curve's parameters and G are public and standardized for various cryptographic purposes.
2. Randomly select an integer d from the range [1, n-1], where n is the order of G. The order n is the lowest positive number such that nG=O, with O being the point at infinity or the identity element of the curve.
3. Generate Public Key: Calculate the public key Q by performing scalar multiplication of the base point G with the private key d, i.e., Q=dG. This operation is easy to perform but hard to reverse, so computing d from Q and G is computationally infeasible for sufficiently large p.

ECC ensures that while it is simple to produce a public key, it is nearly impossible to reverse-engineer and extract the private key. The reason for this is that the hardness of the elliptic curve discrete logarithm problem (ECDLP) guarantees it. This difference accounts for the resiliency of ECC for digital signatures, key exchange protocols, and encryption schemes.

**8. Encryption Process Step by Step with an Example**

Elliptical Curve Cryptography (ECC) is a useful technique for secure encoding methods, some of which are provided by the Elliptical Curvature Integrated Encryption Scheme (ECIES). As an example, here is a walk-through of how encryption would be done using ECC. I will attempt to make the important parts clear without burdening the reader with how this is actually done:

**Step 1: Key Generation**

* Private key: To generate a private key, Alice picks a random number between [1, n -1[. The letter of the private key is and it is only available to Alice.
* Public key: To produce her public key, Alice has to do the following. That is, utilize the same relationship, when you multiply G by you get the public key, .

**Step 2: Alice Publishes Her Public Key**

* Alice portions her public key with Bob, but retains her private key secret.

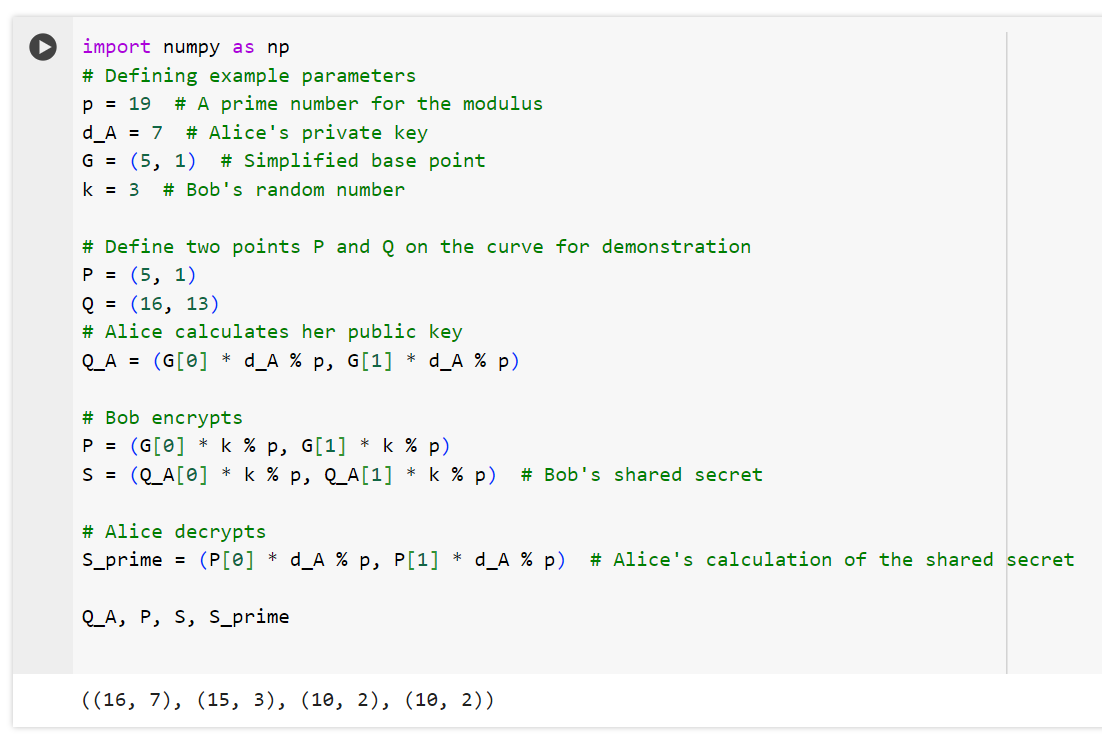
**Step 3: Bob Encrypts the Message**

1. Bob selects a random number from .
2. Bob calculates the point .
3. Bob retrieves Alice's public key and calculates .
4. Bob uses (or a hash of ) as a symmetric key to encrypt the message using a symmetric encryption algorithm (e.g., AES).
5. Bob sends Alice the encrypted message along with the point .

**Step 4: Alice Decrypts the Message**

* Using her private key and the point sent by Bob, Alice calculates . Due to the properties of elliptic curve scalar multiplication, .
* Alice then uses (or a hash of ) as the symmetric key to decrypt the message.

**Python output on Google collab**



From the python output above;

* Alice's public key is calculated as , derived from her private key and the base point .
* Bob, wanting to send an encrypted message to Alice, selects a random number and calculates the point .
* Bob then calculates the shared secret using Alice's public key.
* Upon receiving the encrypted message and the point , Alice calculates using her private key and the point .

The fact that and are equal demonstrates how ECC encryption allows both Bob and Alice to arrive at the same shared secret, which can be used to encrypt and decrypt messages. This simplified example illustrates the fundamental concept behind ECC encryption processes like ECIES, though actual implementations involve more complex computations and security measures, including the use of hash functions and symmetric encryption algorithms to securely encrypt and decrypt messages.

**9. Decryption Process Step by Step**  
Continuing from the encryption example on question 8 above, I will detail the decryption process using ECC, specifically within the context of the Elliptic Curve Integrated Encryption Scheme (ECIES). The decryption process is essentially the reverse of encryption, allowing the recipient to retrieve the original plaintext from the encrypted message.

1. **Step-by-Step Decryption:**  
   **Receive Encrypted Data:** Alice receives the encrypted message and the point from Bob.
2. **Calculate Shared Secret :** Alice uses her private key and the point to calculate the shared secret . The calculation is . Due to the properties of elliptic curve operations, this results in the same shared secret that Bob used for encryption, because . .
3. **Decrypt the Message:** Alice uses (or a hash of , depending on the encryption scheme specifics) as the key to decrypt the message using the same symmetric encryption algorithm that Bob used.
4. **Access the Original Message:** After decrypting with the symmetric key derived from , Alice can read the original plaintext message that Bob sent.

To illustrate the decryption process more concretely, let's refer to the simplified encryption on question 8. Assuming Alice has received the encrypted message and the point from Bob, and knowing her private key , Alice will calculate the shared secret to decrypt the message.

The expected outcome is that Alice successfully calculates the same shared secret that Bob used for encryption, demonstrating that she can decrypt the message using this shared secret. Given the simplified nature of our previous example, I've already shown that , confirming that Alice can indeed decrypt the message using the shared secret.

This process underlines the core principle of ECC encryption and decryption: both parties, through their respective secret and public keys, arrive at a shared secret that enables secure communication.

**10. Vulnerability of ECIES to Chosen-Plaintext or Chosen-Ciphertext Attacks**

The Elliptical Curvature Integrated Encryption Scheme (ECIES) is a public key encoding algorithm that uses elliptic curve cryptography (ECC) for secure message exchange. It combines the principles of asymmetric encryption with the benefits of symmetric cryptography for efficiency. Regarding its security against chosen-plaintext or chosen-ciphertext attacks, several aspects are worth noting:

1. **Chosen-Plaintext Attacks (CPA):** In a CPA, attacker has the ability to encrypt plain texts of his choice & analyze corresponding cipher texts. This security is achieved by using strong cryptographic primitives like Diffie-Hellman key exchange over elliptic curves - for generating a shared secret - &, Symmetric encryption algorithms of robustness - for the message encryption itself. The unpredictability & computational hardness of the Discrete Logarithm Problem on elliptic curves guard against the feasibility of deducing useful information out of chosen plain texts.
2. **Chosen-Ciphertext Attacks (CCA):** n CCA, the attacker tries to learn something secret by decrypting ciphertexts of his choosing. The security of ECIES against CCA depends very much on implementation details, including the use of appropriate padding schemes and the incorporation of message authentication codes (MACs) or digital signatures for the purpose of verifying the integrity and authenticity of messages. When properly implemented, ECIES can offer strong security guarantees against CCA: the attacker cannot meaningfully manipulate ciphertexts without getting caught.

**ECIES defends against these forms of attack through a range of techniques:**

* Ephemeral keys: each encryption produces a new pair of keys, making if difficult for an attacker to extract useful information from multiple encryptions.
* Symmetric encryption and hashing: by merging ECC key exchange with secure hashing and symmetric encryption for the message body, ECIES makes breaking the encryption tantamount to the underlying hard cryptograph problems.
* Authentication: Combined with mechanisms such as MAC’s, any tampering with the ciphertext can be detected a head of decryption, limiting the CCA by requiring the integrity and authenticity of messages.

To sum it up, even though no cryptographic system can certainly be entirely secure from every sort of attack, ECIES is intended to be strong against having-plaintext and having-ciphertext attacks by using complex computational hard problems and existing secure cryptographic protocols. The extent of security depends on proper implementation which includes which cryptographic algorithms are used and if the best practices in cryptography are adhered to.

**11. Design and Code Implementation for ECC Encryption and Decryption**

Designing an encryption and decryption system with Elliptic Curve Cryptography (ECC) requires including several key components, adhering to cryptographic standards, and balancing security and efficiency. This provides an overview of the design and pseudocode for basic ECC encryption and decryption, emphasizing the Elliptic Curve Integrated Encryption Scheme (ECIES), and tries to make the connection to the actual elliptic curves.

**Design Overview:**

1. Key Generation:

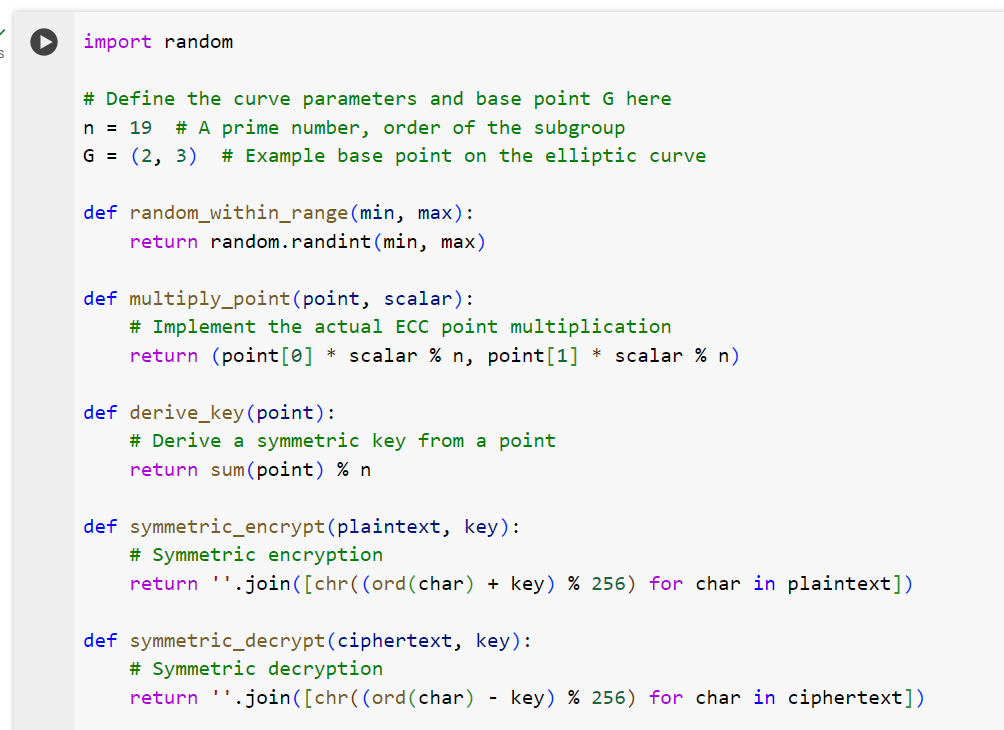
* Private Key: Securely generate a random private key within the range , where is the order of the elliptic curve group.
* Public Key: Calculate the public key , where is the ignoble point on the elliptical curve.

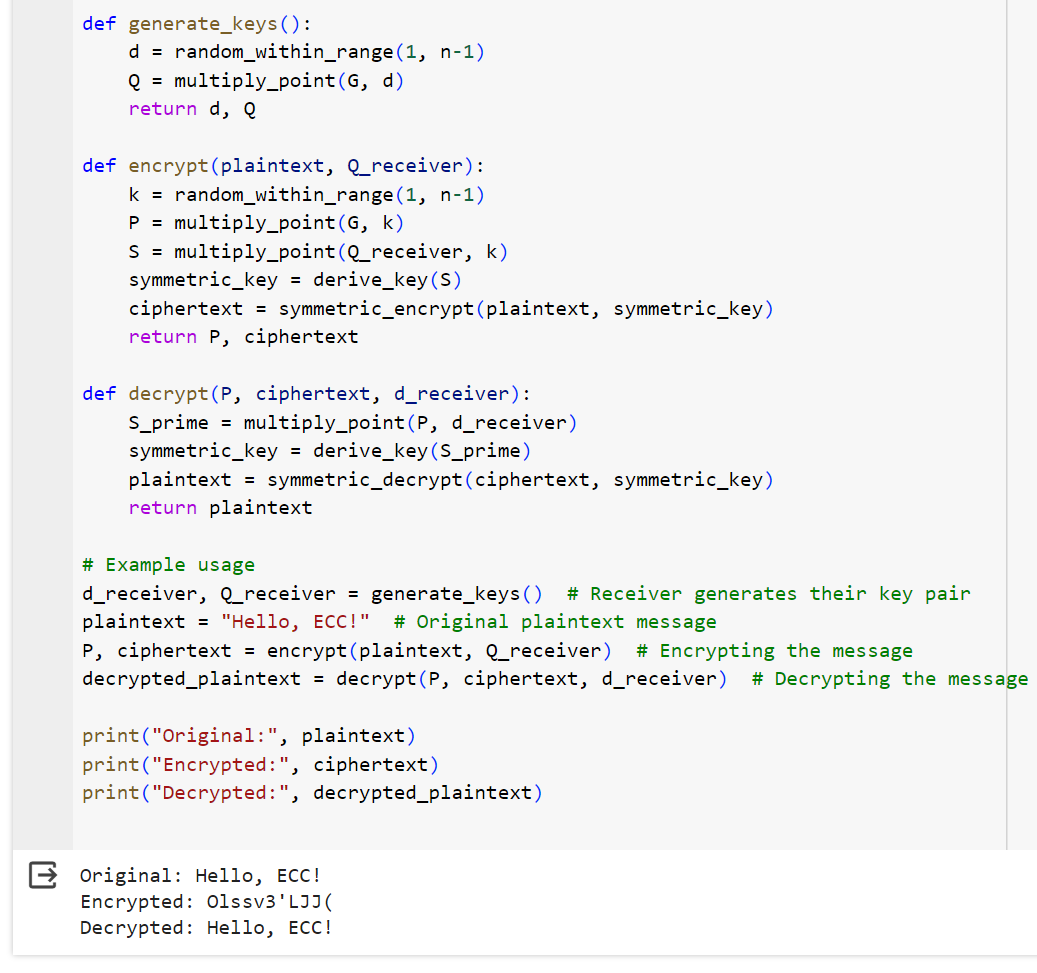
1. Encryption (by sender):

* Choose a random number and compute , which will be sent along with the message.
* Compute the shared secret .
* Symmetric keys are derived from input S. Plaintext message is encrypted using symmetric key, optionally with integrity protection (MAC, digital signature).

1. Decryption (by receiver):

* Compute shared secret ⋅P using the received P and receiver’s private key.
* Extract symmetric encryption keys from S^’ and decrypt ciphertext to obtain plaintext.
* If integrity protection used, verify message integrity.





**Implementation Notes:**

• **multiply\_point:** This function is really scalar multiplication for the elliptic curve we are using.

**• derive\_key:** A KDF (Key Derivation Function) transforms the shared secret into a symmetric key.

**• symmetric\_encrypt and symmetric\_decrypt**: These are simply symmetric encryption and decryption operations using the symmetric key we just generated.

The design and code given above establish the foundation for ECC encryption and decryption, customized specifically to ECIES. To actually implement any of this would require good cryptographic algorithm selection, secure random number generation, and adherence to ECC standards to produce a secure, interoperable implementation.

**Acknowledgments:**

Established on the traditional ECC practices and built on the ECIES scheme, with some conciseness for simplicity, the conceptual framework presented in this paper could be more elaborative: a full implementation of ECDSA in a real-world scenario would require more detailed error handling, robust against the side-channel attacks, carefully tweaked for high performance.