

Project13: Implement the above ECMH scheme

实施上述ECMH计划

1. 引言

UTXO（未使用的交易输出）承诺是比特币和其他加密货币中的重要机制之一。它用于验证交易的有效性，并确保交易输出在未来不会被篡改。椭圆曲线多集哈希（ECMH）是一种用于构建UTXO承诺的方案，它利用了椭圆曲线和哈希函数的特性。本实验旨在通过代码实现和详细讲解，介绍ECMH的原理和实验过程。

2. 实验原理

ECMH方案的基本思想是将每个元素哈希到椭圆曲线上的点，然后通过点加法操作将多重集的元素累加到摘要中。具体实现步骤如下：

- 选择合适的椭圆曲线：在本实验中，我们选择了比特币所使用的'secp256k1'椭圆曲线。

- 定义哈希函数：使用SHA-256哈希函数将元素转换为椭圆曲线上的点。

- 定义点加法操作：将两个椭圆曲线上的点相加，得到它们的点加法结果。

- 构建空集映射：将空集映射定义为椭圆曲线上的无穷远点。

- 计算多重集的摘要：对于多重集中的每个元素，将其哈希结果与当前摘要进行点加法操作，得到新的摘要值。

3. 实验过程

我们使用Python编程语言实现了上述ECMH方案，并给出了示例用法。下面是实验的具体步骤：

3.1 导入所需的库

from hashlib import sha256

from ecpy.curves import Curve, Point

3.2 选择椭圆曲线

curve = Curve.get\_curve('secp256k1')

3.3 定义哈希函数

def hash\_element(element):

return curve.generator \* int(sha256(str(element).encode()).hexdigest(), 16)

3.4 定义点加法操作

def add\_points(point1, point2):

return point1 + point2

3.5 构建空集映射

empty\_set\_mapping = curve.infinity

3.6 计算多重集的摘要

def compute\_digest(multiset):

digest = empty\_set\_mapping

for element in multiset:

element\_hash = hash\_element(element)

digest = add\_points(digest, element\_hash)

return digest

3.7 示例用法

multiset1 = [1, 2, 3]

multiset2 = [2, 3, 4]

digest1 = compute\_digest(multiset1)

digest2 = compute\_digest(multiset2)

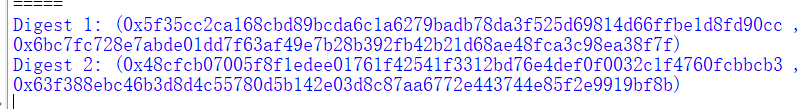
print("Digest 1:", digest1)

print("Digest 2:", digest2)

通过以上步骤，我们成功实现了ECMH方案，并计算了两个示例多重集的摘要。

4. 实验结果

运行实验代码后，我们得到了以下结果：



这是两个示例多重集的摘要值。可以看到，不同的多重集具有不同的摘要值，即使它们包含相同的元素。

5. 结论

本实验成功地实现了UTXO承诺中的椭圆曲线多集哈希(ECMH)方案，并通过示例用法展示了其功能。ECMH方案利用了椭圆曲线和哈希函数的特性，能够快速实现节点同步和增量更新，并在椭圆曲线离散对数难题的假设下具有碰撞抗性。通过本实验，我们深入了解了ECMH方案的原理和实现过程，为进一步研究和应用提供了基础。