实验报告：伪造签名假装你是中本聪

一、引言

椭圆曲线数字签名算法是一种常用的密码学算法，用于确保数据的完整性和认证身份。本实验旨在通过伪造签名的过程，模拟攻击者使用伪造签名假装成中本聪的情景。

二、实验原理

1. 椭圆曲线参数

实验中使用的椭圆曲线参数如下：

- a = 2

- b = 2

- p = 17（模数）

- 曲线上的基点 g(5, 1)

- 阶数 n = 19

2. 生成密钥对和签名

首先，随机选择一个私钥 d，然后根据椭圆曲线点乘法运算生成公钥 pk = d \* g。接着，选择一个随机参数 k，并使用私钥和参数 k 对消息 m 进行签名，得到签名值 (r, s)。

3. 伪造签名攻击过程

为了伪造签名并假装成中本聪，我们需要进行以下步骤：

- 生成两个随机参数 a 和 b

- 使用参数 a 和 b 对基点 g 进行点乘法运算，得到点 Z

- 计算伪造消息的哈希值 e1

- 计算伪造签名的 r1 和 s1

- 调用 `attack` 函数，判断伪造签名是否通过验证

三、实验过程

根据以上原理，我们编写了以下代码进行实验：

```python

import random

import ecdsa

def attack(e, r, s):

global pk, n, g

t = ecdsa.xgcd(s, n) # t = s^(-1) mod n

Z = ecdsa.Add(ecdsa.Multiply((e \* t) % n, g), ecdsa.Multiply((r \* t) % n, pk))

if Z != 0 and Z[0] % n == r:

return True

return False

def satoshi(r, s):

global n, g, pk

a = random.randint(1, n - 1)

b = random.randint(1, n - 1)

Z = ecdsa.Add(ecdsa.Multiply(a, g), ecdsa.Multiply(b, pk))

r1 = Z[0] % n

e1 = (r1 \* a \* ecdsa.gcd(b, n)) % n

s1 = (r1 \* ecdsa.xgcd(b, n)) % n

print('fakemsg:', e1)

print('fakesig:', r1, s1)

if attack(e1, r1, s1):

print('success!')

else:

print('failure!')

# 椭圆曲线参数

a = 2

b = 2

p = 17

x = 5

y = 1

g = [x, y]

n = 19

m = 'Satoshi'

e = hash(m)

d = 7

pk = ecdsa.Multiply(d, g)

k = random.randint(1, n - 1)

r, s = ecdsa.ecdsasignal(m, n, g, d, k)

print("sig", r, s)

if ecdsa.verify(m, n, g, r, s, pk):

print('success')

else:

print('failure')

satoshi(r, s)

```

四、实验结果

实验结果表明，我们成功伪造了一条消息，并生成了伪造的签名。在调用 `attack` 函数时，伪造的签名能够通过验证，输出 "success!"。

五、实验总结

本实验基于椭圆曲线数字签名算法，通过伪造签名的过程，模拟了攻击者伪装成中本聪的情景。实验结果表明，如果签名算法的实现存在漏洞，攻击者可以伪造签名并冒充他人身份进行欺骗。因此，在实际应用中，需要选择安全且经过验证的密码学算法，并确保系统的安全性和完整性。