```
'''''已知椭圆曲线加密 Ep(a,b)参数为
p = 15424654874903
a = 16546484
b = 4548674875
G(6478678675,5636379357093)
私钥为
k = 546768
求公钥 K(x,y)'''
import libnum

#通过二进制分解私钥 K 计算公钥
#依次计算 2G，4G，8G 的值，遇到二进制为 1 的位，就进行结果相加例如 11=1G+2G+8G，但是初始值不好确定，所以加入 first 确定第一个初始值
def main(p,a,b,k,x,y):
    str=bin(k)[2:][::-1]
    result_x,result_y=0,0
    first=True
    #初值设置
    if str[0]=='1':
        result_x,result_y=x,y
        first=False
    #从 2G 开始计算
    for i in range(1,len(str),1):
        x,y=compute(x,y,x,y)
        #二进制为 1 则加到结果上
        if(str[i]=='1'):
            #初值判断
            if first:
                result_x,result_y=x,y
                first=False
            #加到结果上
            else:
                result_x,result_y=compute(result_x,result_y,x,y)
    return result_x,result_y

#计算 a/b mod p 的结果
def div(a,b,p):
    #对 b 求逆
    d=libnum.invmod(b,p)
    return a*d%p

#计算(x1,y1)+(x2,y2)的结果
def compute(x1,y1,x2,y2):
    #根据不同情况计算 lamda 的值
```

```
    if x1==x2 and y1==y2:
        lamda1=(3*(x2**2)+a)%p
        lamda2=(2*y1)%p
    else:
        lamda1=(y2-y1)%p
        lamda2=(x2-x1)%p

    lamda=div(lamda1,lamda2,p)
    x3=(lamda**2-x1-x2)% p
    y3=(lamda*(x1-x3)-y1)% p

    return x3,y3

if __name__=="__main__":
    p = 15424654874903
    a = 16546484
    b = 4548674875
    k = 546768
    x=6478678675
    y=5636379357093
    x,y=main(p,a,b,k,x,y)
    print(x+y)
    print("公钥 K(%d,%d)"%(x,y))
```