本文主要是使用MATLAB演示椭圆曲线加密算法（ECC）的加密/解密过程，内容包括密钥、公钥生成，以及通过加密并解密一个简单数字的过程来描述其使用方法。   
本文实际是对以下两篇文章的一个MATLAB实现，并且提供了两个实用的MATLAB工具函数以便在阅读过程中可以随时检验计算的结果。本文不讲ECC数学原理，但代码中又实际用到其数学规则，所以读者应该结合下面两篇文章阅读本文。（本文更像是对下面第一篇文章内容的一个检验）   
- 《[ECC椭圆曲线详解(有具体实例)](https://www.cnblogs.com/Kalafinaian/p/7392505.html)》   
- 《[Elliptic Curve Cryptography: a gentle introduction](http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/)》

**工具**

**当指定x的值时，计算区间[0,p]内所有y的值。**

这个工具功能很容易理解，通俗点说就是给定x求y,y的取值范围为[0,p]

%file:ECCCal.m

%a,b为椭圆参数，p为质数，x为给定的值

function [ y ] = ECCCal( a,b,p,x )

y=[];

mm = mod(x^3+a\*x+b,p);

index = 1;

for yy = 0:1:p

if mod(yy^2,p) == mm

y(index)=yy;

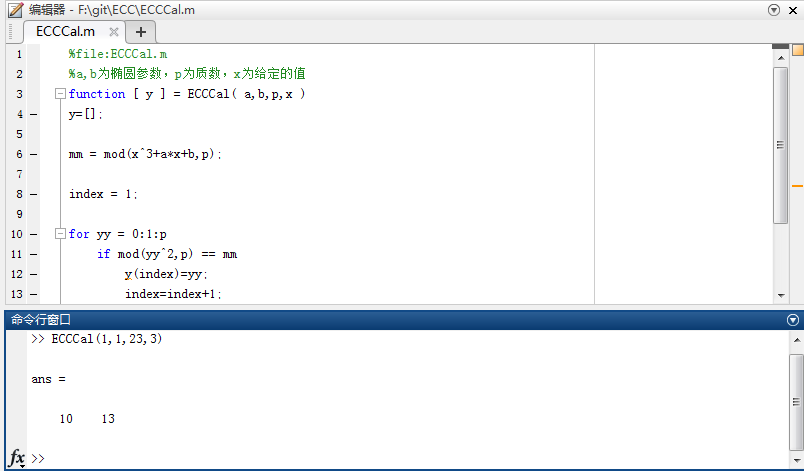
index=index+1;

end

end

end

如计算ECCCal(1,1,23,3)可以得到两个结果 10和13 （有多个结果是正常的不用在意，如果不限定y的范围，理论上是有无数个的）



**计算(0,0)->(p,p)正方形范围内所有满足“椭圆公式“的点**

使用这个工具可以方便我们将该范围内所有满足“椭圆公式”（如果看了上边说的两篇文章就知道这里的椭圆公式可不是指传统意义上的椭圆公式）的点，并将其打印出来。

%file:ECCPlot.m

%a,b分别为椭圆的参数，p为一个质数

function [x,y] = ECCPlot( a,b,p)

x=[];

y=[];

index = 1;

for xr = 0:1:p

mm = mod(xr^3+a\*xr+b ,p);

for yr=0:1:p

if mod(yr^2,p) == mm

x(index)=xr;

y(index)=yr;

index = index+1;

end

end

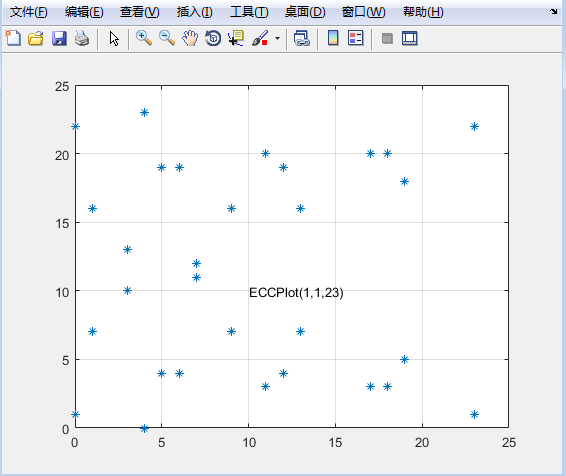
end

plot(x,y,'\*')

hold on

grid on

end

例如我们在MATLAB终端调用ECCPlot(1,1,23)可以看到如下图片   
   
跟推荐的第一篇文章里的图点的坐标位置是一样的

**基本操作方法的MATLAB实现**

因为椭圆曲线加密算法的特殊性我们需要自己实现几个操作方法

**分数的模运算**

我们很容易知道 2对23取模的结果是2，但1212对23的模是多少呢？MATLAB里本身是没有这种计算的方法的（也许是因为我没找到？）所以这里我们需要自己实现

%file:modfrac.m

% n 分子 d 分母 m 模数

function y = modfrac( n,d,m )

n=mod(n,m);

d=mod(d,m);

i=1;

while mod(d\*i,m) ~=1

i=i+1;

end

y=mod(n\*i,m);

end

其实这里还有一个隐藏的陷阱：很多人理解求模就是求除法的余数，但这样是不对的。例如 -2 mod 23 求余数 -2%23=-2但求模mod(-2,23)=21，一定要知道求模运算不能简单的理解为求余数，至少不能用%来计算。

**加法运算**

我们还需要自定义满足椭圆曲线方法的两个点的加法运算，具体规则此处不再赘述了，请查看摘要中提到的两篇文章。

%file:Add.m

%a,b 椭圆参数 p 质数 x1,y1 第一个点的坐标 x2,y2 第二个点的坐标

function [ resx,resy ] = Add( a,b,p,x1,y1,x2,y2 )

if x1==x2 && y1==y2

k=modfrac(3\*x1^2+a,2\*y1,p);

resx = mod(k^2-x1-x2,p);

resy = mod(k\*(x1-resx)-y1,p);

end

if x1==x2 && y1~=y2

resx = inf;

resy = inf;

end

if x1 ~= x2

k=modfrac(y2-y1,x2-x1,p);

resx = mod(k^2-x1-x2,p);

resy = mod(k\*(x1-resx)-y1,p);

end

end

如两点P(3,10)，Q(9,7) 计算P+Q：[x,y]=Add(1,1,23,3,10,9,7)得到结果 x=17,y=20（跟第一篇文章中给的示例的结果是一样的）

**常量乘法**

什么是常量乘法呢？实际上就是N（N为正整数）乖以点P。具体逻辑也需要查看摘要中的文章，如果您阅读的比较详细还会发现第1篇文章给的示例中在求2P的过程中有一处错误。同时也需要说明的是下边MATLAB的实现并没有使用第2篇文章中提到的更快速的算法（为了简单，使用了递归累加的计算方法）

%file:NP.m

%a,b 椭圆参数，p 质数，n表示 n个点P相加也就是n\*P ,x,y 表示P点的横纵坐标

function [resx,resy] = NP( a,b,p,n,x,y )

if n ==1

resx = x;

resy = y;

return;

end

if n>=2

[xsub,ysub]=NP(a,b,p,n-1,x,y);

if xsub==Inf && ysub == Inf

resx=Inf;

resy=Inf;

else

[resx,resy]=Add(a,b,p,x,y,xsub,ysub);

end

end

end

如[x,y]=NP(1,1,23,2,3,10)结果x=7,y=12,[x,y]=NP(1,1,23,3,3,10)结果x=19,y=5

**加密解密过程**

至此我们加密解密需要的所有算法模块就全部足够了。   
流程以摘要中提到的第一篇文章中给的示例为例（我结合本文的代码对其表述进行了修改）

1. Alice选定一条椭圆曲线E，并取椭圆曲线上一点作为基点G 假设选定的椭圆为a=4,b=20,p=29所示示的椭圆，基点G(13,23) , 基点G的阶数n=37（阶数的概念本文没提,你可以理解为在进行常量乘法运算的时候常量的最大值要小于n ）
2. Alice选择一个私有密钥k（k<n）并生成公开密钥K=kG 比如k=25, K= kG = 25G = (14,6）(使用我们的乘法运算函数去测试一下）
3. Alice将E和点K、G传给Bob （ 这没什么说的，E,K,G 共同组成了公钥，需要把公钥发给对方）
4. Bob收到信息后，将待传输的明文编码到上的一点M（编码方法略），并产生一个随机整数r（r<n,n为G的阶数） 假设r=6 要加密的信息为3,因为该信息也需要满足曲线方程E，所以我们很容易能够选取到一点(3,28)（其他的点如(3,1)也是可以的）
5. Bob计算点C1=M+rK和C2=rG
6. Bob将C1、C2传给Alice（容易理解，将加密信息传回给Bob，由Bob解密以知道Alice给他传的是啥）
7. Alice收到信息后，计算C1-kC2得到的结果应该是(3,28) （解密过程）

至此椭圆曲线加密、解密的整个流程就结束了。将其写成一个MATLAB脚本 如下：

%file:ECC.m

%演示曲线加密算法加/解密过程

a=4;

b=20;

p=29;

GX=13;

GY=23;

k=25;

[KX,KY]=NP(a,b,p,k,GX,GY)

r=6

MX = 3

MY = 28

[rKX,rKY] = NP(a,b,p,r,KX,KY)

[C1X,C1Y]=Add(a,b,p,MX,MY,rKX,rKY)

[C2X,C2Y]=NP(a,b,p,r,GX,GY)

[kC2X,kC2Y]=NP(a,b,p,k,C2X,C2Y);

kC2Y=mod(-1\*kC2Y,p)

[resx,resy]=Add(a,b,p,C1X,C1Y,kC2X,kC2Y)

我们执行一下可以看到输出结果 resx=3,resy=28这里的3正是Bob给Alice传的数字内容，其他人因为没有私钥k所以无法像Alice那样简单地还原出数字3

**总结**

惯例文章最后总是要说点什么，我就说说自己的感受吧。   
椭圆加密算法看似简单（仅仅通过一个加法运算和一个常量乘法运算便完成了加密解密算法）但其背后隐藏了很深的数学知识。本文仅仅是起演示功能，真正在实际生产中所使用的实现方式以及加密解密过程还需要进行深入地学习。

--------------------- 本文来自 alphags 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/alphags/article/details/79660819?utm\_source=copy