The Oil and Vinegar Method

William J Buchanan [1][2]

[1] Blockpass ID Lab, Edinburgh Napier University, Edinburgh, UK

[2] Asecuritysite.com

Abstract

Public key encryption methods are often used to create a digital signature, and where Bob has a public key and a private key. In order to prove his identity, he will encrypt something related to the message with his private key, and which can then be checked with his public key. The main current methods of public-key encryption include RSA and ECC (Elliptic Curve Cryptography), and which involve computationally difficult operations. But these operations have not been proven to be hard in an era of quantum computers. One well-known hard problem is the solving of quadratic equations with m equations with n variables. This is a known NP-hard problem, even in a world of quantum computers. These can be used as post-quantum signature schemes and which involve multivariate equations. In order to understand these methods, this paper outlines a simple example of implementing the oil and vinegar method, and where we have a number of unknown oil variables and a number of known vinegar variables, and where the vinegar variables help convert the hard problem into an easy one.

Keywords: Post Quantum Cryptography, Oil and Vinegar

1. Introduction

In a post-quantum world, the existing public key methods of discrete logs (ElGamal), RSA and Elliptic Curve can be broken using Shor's algorithm [1]. Overall, discrete logarithm, integer factorization and elliptic curve methods are not provably hard problems and are just applied in the current era as they as difficult problems with existing computer systems.

Public key encryption is all about setting up a trapdoor function, where someone who knows a given secret will fall through a function. As much as possible, we need a difficult problem to solve, and one which is known to be hard to solve. Knowing a secret allows the hard problem to become simple. For example, Bob might have a secret value of 6, and then use a secret multiplier of 5. His cipher method can then be to multiply a message by his secret to give:

$$Cipher = 6 \times 5 = 30 \tag{1}$$

The trap door is then the inverse of the secret: $\frac{1}{5}$. If you know that, you can now multiply the Cipher by the inverse value. But in this case, the inverse of five is easy to find, so we need tougher ways.

In RSA, we select two prime numbers (p and q) and then determine the modulus (N) and which is the multiplication of p and q. We then encrypt a message of m with $C = M^e \pmod N$ and decrypt with $M = C^d \pmod N$. For this we pick a value of e which does not share a factor with ϕ, and where:

$$\phi = (p-1) \times (q-1) \tag{2}$$

To determine d we must solve this:

$$d \times e \pmod \phi = 1 \tag{3}$$

For this we need to perform an inverse modulus operation:

$$d = e^{-1} \pmod \phi \tag{4}$$

While this method has worked well for over 40 years, it is now at risk in an era of quantum computers. In this paper, we will investigate a provable hard problem and which can have a backdoor applied.

The method outlined in this paper involves taking a hard problem to solve and then applying a trap door. If we know a little secret, the hard problem becomes easy. Overall, quantum computers will be able to break our existing public key methods, such as discrete logs, RSA and elliptic curve. And so NIST has created a Post Quantum Cryptography (PQC) competition, and one of the methods is known as Rainbow [2]. This is known as the Oil and Vinegar method [3] and uses multivariate cryptography with an added trapdoor.

2. Method

With multivariate cryptography, we have n variables within polynomial equations. For example if we have four variables (w, x, y, z) and an order of two, we could have [4]:

$$w^2 + 4wx + 3x^2 + 2wy - 4wz + 2wx + 6xz = 387 \tag{5}$$

In this case I know that the solution is $w = 7$, $x = 4$, $y = 5$ and $z = 6$. For a matrix form we could represent this as:

$$M = \begin{pmatrix} w & x & y & z \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 & 1 \\ 2 & 3 & 3 & -2 \\ 1 & 3 & 0 & 0 \\ 1 & -2 & 0 & 0 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} \tag{6}$$

This matrix has a trapdoor and where we define our vinegar and oil variables. The vinegar variables are secret and which we will only know, and the oil ones will be discovered if we know the vinegar variables. The trap door is that we do not let the oil variables mix, and where they do mix in the matrix, we will have zeros (the trap door) - as illustrated in Figure 1.

In order to constrain the values we get, we normally perform $\pmod p$ operation and where p is a prime number. This is known as finite field, and are numbers are always constrained between 0 and $p - 1$. For example if we select $p = 97$ we have:

$$w^2 + 4wx + 3x^2 + 2wy - 4wz + 2wx + 6xz = 5 \pmod{97} \tag{7}$$

Now there are multiple solutions to this now for w, x, y, z, so we define n multivariate polynomials. For example:

$$w^2 + 4wx + 3x^2 + 2wy - 4xz + 2wx + 6xy = 96 \pmod{97} \tag{8}$$

$$5w^2 + 3wx + 3x^2 + 4wy - xz + 8wx + 4xy = 36 \pmod{97} \tag{9}$$

$$4w^2 + 5wx + 3x^2 + 2wy - 5xz + 3wx + 6xy = 95 \pmod{97} \tag{10}$$

$$6w^2 + 7wx + 4x^2 + 2wy - 8xz + 2wx + 9xy = 17 \pmod{97} \tag{11}$$

Now we have vinegar variables of w and x and oil variables of y and z. If we know w and x it will now become easy to determine oil variables. For example if $w = 7$ and $x = 4$, we get:

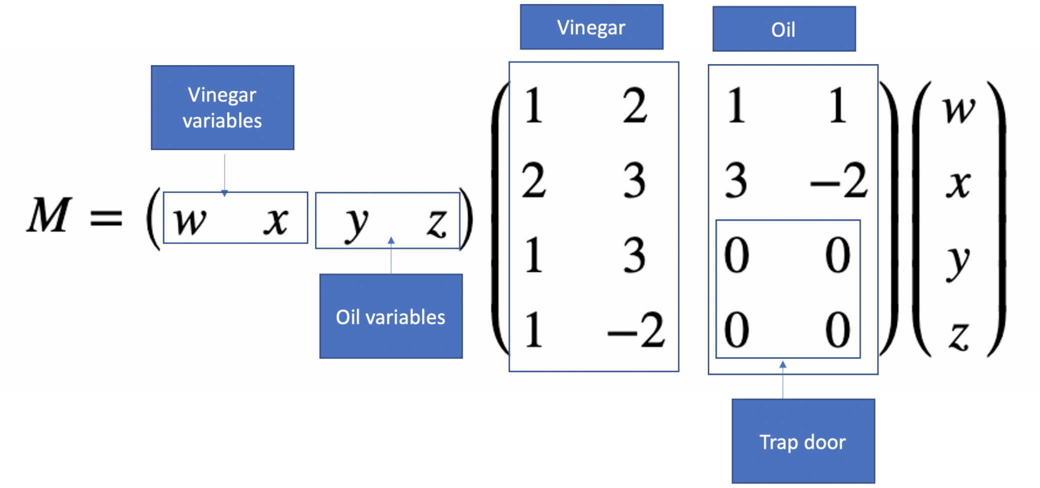

Figure 1: The trapdoor

$$49 + 112 + 48 + 14y - 16z + 14z + 24y = 96 \pmod{97} \tag{12}$$

$$245 + 84 + 48 + 28y - 4z + 56z + 16y = 36 \pmod{97} \tag{13}$$

and gives:

$$209 + 38y - 2z = 96 \pmod{97} \tag{14}$$

$$377 + 44y + 52z = 36 \pmod{97} \tag{15}$$

and gives:

$$38y + 95z = -113 \pmod{97} \tag{16}$$

$$44y + 52z = -341 \pmod{97} \tag{17}$$

and gives:

$$38y + 95z = 81 \pmod{97} \tag{18}$$

$$44y + 52z = 47 \pmod{97} \tag{19}$$

In a matrix form this becomes:

$$\begin{pmatrix} 38 & 95 \\ 44 & 52 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 81 \\ 47 \end{pmatrix} \pmod{97} \quad (20)$$

and we can solve for y and z with:

$$\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 38 & 95 \\ 44 & 52 \end{pmatrix}^{-1} \begin{pmatrix} 81 \\ 47 \end{pmatrix} \pmod{97} \quad (21)$$

We can now easily solve this to get $y = 5$ and $z = 6$.

3. Coding

In the code we need to perform an inverse of the matrix with a modulo operation [5][6]:

```
import numpy as np
from inv import modMatInv
import sys

p=97

w=7
x=4
def printM(M):
  rtn = ""+str(M[0][0])+"w^2 + "+str(M[0][1]+M[1][0])+"wx + "+str(M[1][1])+"
      x^2 + "+str(M[0][2]+M[2][0])+"wy + "+str(M[1][3]+M[3][1])+"xz + " +str
      (M[0][3]+M[3][0])+"wz + "+str(M[1][2]+M[2][1])+"xy"
  return rtn

def revealM(M,w,x):
  rtn = ""+str(M[0][0]*w*w) +" + "+str((M[0][1]+M[1][0])*w*x)+" + "+str(M
      [1][1]*x*x)+" + "+str((M[0][2]+M[2][0])*w)+"y + "+str((M[1][3]+M
      [3][1])*x)+"z + " +str((M[0][3]+M[3][0])*w)+"z + "+str((M[1][2]+M
      [2][1])*x)+"y"
  return rtn

M0=[[1,2,1,1],[2,3,3,-2],[1,3,0,0],[1,-2,0,0]]
M1=[[5,2,1,2],[1,3,2,2],[3,2,0,0],[6,-3,0,0]]
M2=[[4,2,2,1],[3,3,3,-2],[0,3,0,0],[2,-3,0,0]]
M3=[[6,2,1,1],[5,4,3,-3],[1,6,0,0],[1,-5,0,0]]

print ("p=",p)

print ("\nM0:",M0)
print ("M1:",M1)
print ("M2:",M2)
print ("M3:",M3)

# res = m . M0 . m^{-1}
res0=96
res1=36
res2=95
res3=17

a=(res0-(M0[0][0]*(w*w)+(M0[0][1]+M0[1][0])*w*x + (M0[1][1])*x*x ) ) %p
```

```
b=(res1-(M1[0][0]*(w*w)+(M1[0][1]+M1[1][0])*w*x + (M1[1][1])*x*x ) ) %p

factor1=( ((M0[0][2]+M0[2][0])*w) +  ((M0[1][2]+M0[2][1]) *x) )  %p
factor2=( ((M0[0][3]+M0[3][0])*w) + ((M0[1][3]+M0[3][1]) *x) )  %p
factor3=( (M1[0][2]+M1[2][0])*w) +  ((M1[1][2]+M1[2][1]) *x)  %p
factor4=( ((M1[0][3]+M1[3][0])*w) + ((M1[1][3]+M1[3][1])*x))  %p

print ("w=",w)
print ("x=",x)
print (printM(M0))
print (printM(M1))
print ()
print (revealM(M0,w,x))
print (revealM(M1,w,x))
print ()
print (str(factor1)+"y+"+str(factor2)+"z="+str(a))
print (str(factor3)+"y+"+str(factor4)+"z="+str(b))
print ()

A = np.array([[factor1,factor2], [factor3,factor4]])
B = np.array([a,b])

invA = modMatInv(A,p)

print (invA)

res = np.dot(invA,B) % p

print ("y=",res[0])
print ("z=",res[1])
```

A sample run gives:

```
p= 97

M0: [[1, 2, 1, 1], [2, 3, 3, -2], [1, 3, 0, 0], [1, -2, 0, 0]]
M1: [[5, 2, 1, 2], [1, 3, 2, 2], [3, 2, 0, 0], [6, -3, 0, 0]]
M2: [[4, 2, 2, 1], [3, 3, 3, -2], [0, 3, 0, 0], [2, -3, 0, 0]]
M3: [[6, 2, 1, 1], [5, 4, 3, -3], [1, 6, 0, 0], [1, -5, 0, 0]]
w= 7
x= 4
1w^2 + 4wx + 3x^2 + 2wy + -4xz + 2wz + 6xy
5w^2 + 3wx + 3x^2 + 4wy + -1xz + 8wz + 4xy

49 + 112 + 48 + 14y + -16z + 14z + 24y
245 + 84 + 48 + 28y + -4z + 56z + 16y

38y+95z=81
44y+52z=47

Inverse matrix:
 [[63. 36.]
 [81.  5.]]
y= 5.0
z= 6.0
```

4. Conclusions

The Oil and Vinegar method provides us with a hard problem within a post-quantum work, and where we can add a trapdoor.

References

[1] W. Buchanan and A. Woodward, "Will quantum computers be the end of public key encryption?" *Journal of Cyber Security Technology*, vol. 1, no. 1, pp. 1–22, 2017.

[2] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *International Conference on Applied Cryptography and Network Security*. Springer, 2005, pp. 164–175.

[3] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced oil and vinegar signature schemes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 206–222.

[4] B. Minuad, "Code-based, multivariate and hash-based cryptography," https://www.di.ens.fr/brice.minaud/slides/Qhub-2018.pdf, 2018.

[5] W. J. Buchanan, "Simple coding of oil and vinegar," https://repl.it/@billbuchanan/ov, 2020.

[6] ——, "The oil and vinegar method," https://asecuritysite.com/encryption/multiv2, 2020.