

Bit SKV

Bit SRL
Equipo del SLTA-verse

1 Descripción

SKV es un método de rápida autenticación de claves de instalación utilizando multiplicación de matrices, utilizando 4 matrices (el formato de encodeo de las matrices todavía está por definirse).

El cliente de SLTA tendrá (en algún formato no directamente visible) 4 matrices, A_3, B_3, C_2, D_2 donde $A \times B = M1$ y $C \times D = M2$, donde $M1$ y $M2$ se describen en las siguientes secciones.

2 M2

M2 es una matriz de tamaño 2×2 con los siguientes valores:

$$\begin{bmatrix} S & T \\ L & A \end{bmatrix} \xrightarrow{\text{letras a números}} \begin{bmatrix} 19 & 20 \\ 12 & 1 \end{bmatrix}$$

Para llegar a este resultado, necesitaremos 2 matrices C_2, D_2 / $C \times D = M2$.

Por ejemplo, despejando del siguiente modo podemos llegar a valores ejemplo de C y D.

2.1 M2: C y D ejemplares

Empezamos declarando la ecuación

$$C \times D = M2$$

Remplazamos M2 por su valor, y C,D por matrices con variables

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} j & k \\ l & m \end{bmatrix} = \begin{bmatrix} 19 & 20 \\ 12 & 1 \end{bmatrix}$$

Damos a la matriz C valores para poder despejar la matriz D

$$\begin{bmatrix} 5 & 12 \\ 8 & 0 \end{bmatrix} \times \begin{bmatrix} j & k \\ l & m \end{bmatrix} = \begin{bmatrix} 19 & 20 \\ 12 & 1 \end{bmatrix}$$

Definimos las ecuaciones que se están realizando dentro de esta multiplicación (o sea, los dot product entre las filas de C y las columnas de D)

$$\begin{aligned} M2_{11} &= 5.j + 12.l &= 19 \\ M2_{12} &= 5.k + 12.m &= 20 \\ M2_{21} &= 8.j + 0.l &= 12 \\ M2_{22} &= 8.j + 0.m &= 1 \end{aligned}$$

2.2 M2: primer sistema de ecuaciones

Tomamos las ecuaciones de la columna 1 en un sistema de ecuaciones (ya que tienen las mismas variables)

$$\begin{cases} 5j + 12l = 19 \\ 8j = 12 \end{cases}$$

$$1. \ 8j = 12 \rightarrow \boxed{j = \frac{3}{2}}$$

$$2. \ 5.\frac{3}{2} + 12.l = 19 \rightarrow 12l = 19 - \frac{15}{2} = \frac{23}{2} \rightarrow \boxed{l = \frac{23}{24}}$$

2.3 M2: segundo sistema de ecuaciones

Tomamos las ecuaciones de la columna 2 en otro sistema de ecuaciones

$$\begin{cases} 5k + 12m = 20 \\ 8k = 1 \end{cases}$$

$$1. \ 8k = 1 \rightarrow \boxed{k = \frac{1}{8}}$$

$$2. \ 5 \cdot \frac{1}{8} + 12m = 20 \rightarrow 12m = \frac{160-5}{8} \rightarrow \frac{96m}{8} = \frac{155}{8} \rightarrow \boxed{m = \frac{155}{96}}$$

2.4 M2: Resultado

Tras estos sistemas de ecuaciones, tenemos

$$C = \begin{bmatrix} 5 & 12 \\ 8 & 0 \end{bmatrix}, D = \begin{bmatrix} \frac{3}{2} & \frac{1}{8} \\ \frac{23}{24} & \frac{155}{96} \end{bmatrix}$$

Verificamos que $C \times D$ sea igual a M2

$$\begin{aligned} (C \times D)_{11} &= 5 \cdot \frac{3}{2} + 12 \cdot \frac{23}{24} &= \frac{456}{24} &= 19 \\ (C \times D)_{12} &= \frac{5}{8} + 12 \cdot \frac{155}{96} &= \frac{1920}{96} &= 20 \\ (C \times D)_{21} &= 8 \cdot \frac{3}{2} + 0 \cdot \frac{23}{24} &= &= 12 \\ (C \times D)_{22} &= 8 \cdot \frac{1}{8} + 0 \cdot \frac{155}{96} &= &= 1 \end{aligned}$$

$$= \begin{bmatrix} 19 & 20 \\ 12 & 1 \end{bmatrix}$$

Lo cual significa que $\{C, D\}$ es parte de una clave válida para el sistema de Bit SKV.

3 M1

M1 es una matriz de tamaño 3×3 con los siguientes valores:

$$\begin{bmatrix} B & S & S \\ I & R & K \\ T & L & V \end{bmatrix} \xrightarrow{\text{letras a números}} \begin{bmatrix} 2 & 19 & 19 \\ 9 & 18 & 11 \\ 20 & 11 & 22 \end{bmatrix}$$

Para llegar a este resultado, necesitaremos 2 matrices $A_3, B_3/A \times B = M1$.

Por ejemplo, despejando del siguiente modo podemos llegar a valores ejemplo de A y B.

3.1 M1: A y B ejemplares

Empezamos declarando la ecuación

$$A \times B = M1$$

Remplazamos M1 por su valor, y A,B por Matrices con variables

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \times \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix} = \begin{bmatrix} 2 & 19 & 19 \\ 9 & 18 & 11 \\ 20 & 11 & 22 \end{bmatrix}$$

Damos a la matriz A valores para poder despejar la matriz B

$$\begin{bmatrix} 1 & 3 & 0 \\ 5 & 8 & 12 \\ 9 & 4 & 10 \end{bmatrix} \times \begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix} = \begin{bmatrix} 2 & 19 & 19 \\ 9 & 18 & 11 \\ 20 & 11 & 22 \end{bmatrix}$$

Definimos las ecuaciones que se están realizando dentro de esta multiplicación

$$\begin{aligned} M1_{11} &= 1.j + 3.m + 0.p &= 2 \\ M1_{12} &= 1.k + 3.n + 0.q &= 19 \\ M1_{13} &= 1.l + 3.o + 0.r &= 19 \\ M1_{21} &= 5.j + 8.m + 12.p &= 9 \\ M1_{22} &= 5.k + 8.n + 12.q &= 18 \\ M1_{23} &= 5.l + 8.o + 12.r &= 11 \\ M1_{31} &= 9.j + 4.m + 10.p &= 20 \\ M1_{32} &= 9.k + 4.n + 10.q &= 11 \\ M1_{33} &= 9.l + 4.o + 10.r &= 22 \end{aligned}$$

3.2 M1: primer sistema de ecuaciones

Separamos las ecuaciones para $M1_{11}, M1_{21}, M1_{31}$ en un sistema de ecuaciones ya que tienen las mismas variables:

$$\begin{cases} 1.j + 3.m &= 2 \\ 5.j + 8.m + 12.p &= 9(-10) \\ 9.j + 4.m + 10.p &= 20(12) \end{cases}$$

Multiplicamos la 2da y 3ra ecuación

$$\begin{cases} -50.j - 80.m - 120.p &= -90 \\ 108.j + 48.m + 120.p &= 240 \end{cases}$$
$$58.j - 32.m = 150$$

Incluimos la ecuación resultante en un sistema de ecuaciones 2x2 junto con la 1ra ecuación del sistema anterior:

$$\begin{cases} 58.j - 32.m &= 150(1) \\ j - 3.m &= 2(-58) \end{cases}$$

Multiplicamos las ecuaciones y reducimos:

$$\begin{aligned} 58.j - 32.m &= 150 \\ -58.j - 174.m &= -116 \\ -206.m &= 34 \\ m &= \frac{-17}{103} \end{aligned}$$

Reusamos la primera ecuación $j + 3.m = 2$ para despejar j

$$\begin{aligned} j + 3.\left(\frac{-17}{103}\right) &= 2 \\ j &= 2 + \frac{51}{103} = \frac{257}{103} \end{aligned}$$

Reusamos la segunda ecuación $5.j + 8.m + 12.p = 9$ para despejar p

$$\begin{aligned} 5.\left(\frac{257}{103}\right) + 8.\left(\frac{-17}{103}\right) + 12.p &= 9 \\ \frac{1285 - 136 - 927}{103} &= -12.p \\ 12.p &= \frac{-222}{103} \\ p &= \frac{-37}{206} \end{aligned}$$

3.3 M1: segundo sistema de ecuaciones

Separamos las ecuaciones para $M1_{12}, M1_{22}, M1_{32}$ en un sistema de ecuaciones

$$\begin{cases} 1.k + 3.n &= 19 \\ 5.k + 8.n + 12.q &= 18(-10) \\ 9.k + 4.n + 10.q &= 11(12) \end{cases}$$

Multiplicamos las ecuaciones

$$\begin{cases} -50.k - 80.n - 120.q &= -180 \\ 108.k + 48.n + 120.q &= 132 \\ 58.k - 32.n &= -48 \end{cases}$$

Incluimos la ecuación resultante en un sistema de ecuaciones 2x2 junto con la 1ra ecuación del sistema anterior:

$$\begin{cases} 58.k - 32.n &= -48(-1) \\ k - 3.n &= 19(58) \end{cases}$$

Multiplicamos las ecuaciones y reducimos:

$$\begin{aligned} -58.k + 32.n &= 48 \\ 58.k + 174.n &= 1102 \\ 206.n &= 1150 \\ n &= \frac{575}{103} \end{aligned}$$

Reusamos la primera ecuación $k + 3.n = 19$ para despejar k

$$\begin{aligned} n + 3.(\frac{575}{103}) &= 19 \\ n &= 19 - \frac{1725}{103} = \frac{232}{103} \end{aligned}$$

Reusamos la segunda ecuación $5.k + 8.n + 12.q$ para despejar q

$$\begin{aligned} 5.(\frac{232}{103}) + 8.(\frac{575}{103}) + 12.q &= 18 \\ \frac{1160 + 4600 - 1854}{103} &= -12.q \\ q &= \frac{-3906}{1236} \\ q &= \frac{-651}{206} \end{aligned}$$

3.4 M1: Tercer sistema de ecuaciones

Separamos las ecuaciones para M_{13}, M_{123}, M_{133} en un sistema de ecuaciones

$$\begin{cases} 1.l + 3.o = 19 \\ 5.l + 8.o + 12.r = 11(-10) \\ 9.l + 4.o + 10.r = 22(12) \end{cases}$$

Multiplicamos las ecuaciones

$$\begin{cases} -50.l - 80.o - 120.r = -110 \\ 108.l + 48.o + 120.r = 264 \\ 58.l - 32.o = 154 \end{cases}$$

Incluimos la ecuación resultante en un sistema de ecuaciones 2x2 junto con la 1ra ecuación del sistema anterior:

$$\begin{cases} 58.l - 32.o = 154 \\ l - 3.o = 19(-58) \end{cases}$$

Multiplicamos la ecuación y reducimos:

$$\begin{aligned} 58.l - 32.o &= 154 \\ -58.l - 174.o &= -1102 \\ -206.o &= -948 \\ o &= \frac{474}{103} \end{aligned}$$

Reusamos la primera ecuación $l + 3.o = 19$ para despejar l

$$\begin{aligned} l + 3.\left(\frac{474}{103}\right) &= 19 \\ l &= 19 - \frac{1422}{103} = \frac{535}{103} \end{aligned}$$

Reusamos la segunda ecuación $5.l + 8.o + 12.r$ para despejar r

$$\begin{aligned} 5.\left(\frac{535}{103}\right) + 8.\left(\frac{474}{103}\right) + 12.r &= 11 \\ \frac{2675 + 3792 - 1133}{103} &= -12.r \\ r &= \frac{-5334}{1236} \\ r &= \frac{-889}{206} \end{aligned}$$

3.5 M1: Resultado

Tras estos sistemas de ecuaciones, tenemos

$$A = \begin{bmatrix} 1 & 3 & 0 \\ 5 & 8 & 12 \\ 9 & 4 & 10 \end{bmatrix}, B = \begin{bmatrix} \frac{257}{103} & \frac{232}{103} & \frac{535}{103} \\ -17 & 575 & 474 \\ \frac{103}{-37} & \frac{103}{-651} & \frac{103}{-889} \end{bmatrix}$$

4 Verificación de claves mediante M1 y M2

Bit SKV es un algoritmo para claves de autenticación de los productos de Bit SRL, los cuales tendrían como matrices A,C o B,D una tabla de datos que permita identificar al cliente y la otra matriz despejada a partir de la misma. Posteriormente, ambas se expresarían bajo la siguiente estructura de $((3 * 3 * 2 * 2) + (2 * 2 * 2 * 2)) * 2 = 104$ bytes:

Cada número se representa con 4 bytes, 2 para su numerador (firmado) y 2 para su denominador (no firmado).

Los primeros 36 bytes representan los números de A, donde $R_i = A_{(i \bmod 3)+1, (i \bmod 3)+1}$, o sea los números de la misma fila se almacenan conjuntamente. Visualmente, esto parece

A_{11}	A_{12}	A_{13}	A_{21}	A_{22}	\dots	A_{32}	A_{33}
----------	----------	----------	----------	----------	---------	----------	----------

Los próximos 36 bytes representan los números de B del mismo modo. Los últimos 32 bytes representan los números de C y D del mismo modo.

Estos 104 bytes son codificados mediante *base64*, un sistema de codificación binaria que utiliza caracteres del abecedario + dígitos + algunos caracteres especiales para codificar de una manera legible información en formato binario. Por ejemplo, nuestros ABCD de ejemplo en formato b64 serían el string (sin rotura de líneas)

AQABAAMAAQAAAAEABQABAAgAAQAMAAEACQABAAQAAQAKAAEAAQFnAOgAZwAXAmcA7/9nAD8C
ZwDaAWcA2//OAHX9zgCH/M4ABQABAAwAAQAIAAEAAAABAAMAAgABAAGfWAYAJsAYAA=

4.1 Programa ejemplo usando estas matrices

Un programa ejemplo, que encodea y decodea estas matrices (y muestra el resultado de $A*B$ y $C*D$) es

```
1 Imports System
2 Imports System.Linq
3 Imports Fractions
4
5 Public Structure Matrix
6     Public Values(,) as Fraction
7     Public ReadOnly Property Rows as Integer
8         Get
9             Return Values.GetLength(0)
10        End Get
11    End Property
12
13    Public ReadOnly Property Columns as Integer
14        Get
15            Return Values.GetLength(1)
16        End Get
17    End Property
18
19    Public Sub New(mat as IEnumerable(Of IEnumerable(Of
20        Fraction)))
21        ReDim Values(mat.Count-1, mat(0).Count-1)
22        For i = 0 to mat.Count-1
23            For j = 0 to mat(0).Count-1
24                Values(i,j) = mat(i)(j)
25            Next
26        Next
27    End Sub
28
29    Public Overrides Function ToString() As String
30        Dim str = ""
31        For y = 0 to Columns-1
32            For x = 0 to Rows-1
33                str += Values(y,x).ToString()
34                str += vbTab
35            Next
36            str += vbNewLine
```

```

36         Next
37         Return str
38     End Function
39
40     Public Shared Operator *(ByVal m1 as Matrix, ByVal
      m2 as Matrix) as Matrix
41         If m1.Columns <> m2.Rows Then
42             Throw New Exception("m1.Columns_must_be_="
      _m2.Rows")
43         End If
44         Dim values as New List(Of List(Of Fraction))
45         For i = 0 to m1.Rows-1
46             Dim row = new List(Of Fraction)
47             values.Add(row)
48             For j = 0 to m2.Columns-1
49                 Dim val as Fraction = 0
50                 For k = 0 to m2.Rows-1
51                     val += (m1.Values(i,k) * m2.
      Values(k,j))
52                 Next
53                 row.Add(val)
54             Next
55         Next
56         Return New Matrix(values)
57     End Operator
58 End Structure
59
60 Module Program
61     Sub Encode()
62         Dim input = Console.ReadLine
63         Dim values_a = input.Split(";").Select(Function(
      x) x.Split(",").Select(Function(i) i.Split("/"
      "))).ToList()
64         input = Console.ReadLine
65         Dim values_b = input.Split(";").Select(Function(
      x) x.Split(",").Select(Function(i) i.Split("/"
      "))).ToList()
66         input = Console.ReadLine
67         Dim values_c = input.Split(";").Select(Function(
      x) x.Split(",").Select(Function(i) i.Split("/

```

```

        "))) .ToList()
68      input = Console.ReadLine
69      Dim values_d = input.Split(";").Select(Function(
        x) x.Split(",").Select(Function(i) i.Split("/
        "))) .ToList()
70      Dim bytevals(103) as Byte
71      For x = 0 to 2
72          For y = 0 to 2
73              Try
74                  Dim num_d() as Byte = BitConverter.
                      GetBytes(CType(Integer.Parse(values_a
                      (y)(x)(0)), Int16))
75                  bytevals((y*3*4)+(x*4)) = num_d(0)
76                  bytevals((y*3*4)+(x*4)+1) = num_d(1)
77                  Dim div_d() as Byte = BitConverter.
                      GetBytes(CType(Integer.Parse(values_a
                      (y)(x)(1)), Int16))
78                  bytevals((y*3*4)+(x*4)+2) = div_d(0)
79                  bytevals((y*3*4)+(x*4)+3) = div_d(1)
80              Catch e as Exception
81                  Console.WriteLine(e)
82                  Console.WriteLine(x)
83                  Console.WriteLine(y)
84              End Try
85          Next
86      Next
87      For x = 0 to 2
88          For y = 0 to 2
89              Dim num_d() as Byte = BitConverter.
                      GetBytes(CType(Integer.Parse(values_b
                      (y)(x)(0)), Int16))
90                  bytevals((y*4*3)+(x*4)+36) = num_d(0)
91                  bytevals((y*4*3)+(x*4)+37) = num_d(1)
92                  Dim div_d() as Byte = BitConverter.
                      GetBytes(CType(Integer.Parse(values_b
                      (y)(x)(1)), Int16))
93                  bytevals((y*4*3)+(x*4)+38) = div_d(0)
94                  bytevals((y*4*3)+(x*4)+39) = div_d(1)
95              Next
96      Next

```

```

97      For x = 0 to 1
98          For y = 0 to 1
99              Dim num_d() as Byte = BitConverter.
                  GetBytes(CType(Integer.Parse(values_c
100                  (y)(x)(0)), Int16))
101                  bytevals((y*4*2)+(x*4)+72) = num_d(0)
102                  bytevals((y*4*2)+(x*4)+73) = num_d(1)
103              Dim div_d() as Byte = BitConverter.
                  GetBytes(CType(Integer.Parse(values_c
104                  (y)(x)(1)), Int16))
105                  bytevals((y*4*2)+(x*4)+74) = div_d(0)
106                  bytevals((y*4*2)+(x*4)+75) = div_d(1)
107          Next
108      Next
109      For x = 0 to 1
110          For y = 0 to 1
111              Dim num_d() as Byte = BitConverter.
                  GetBytes(CType(Integer.Parse(values_d
112                  (y)(x)(0)), Int16))
113                  bytevals((y*4*2)+(x*4)+88) = num_d(0)
114                  bytevals((y*4*2)+(x*4)+89) = num_d(1)
115              Dim div_d() as Byte = BitConverter.
                  GetBytes(CType(Integer.Parse(values_d
116                  (y)(x)(1)), Int16))
117                  bytevals((y*4*2)+(x*4)+90) = div_d(0)
118                  bytevals((y*4*2)+(x*4)+91) = div_d(1)
119          Next
120      Next
121      Console.WriteLine(System.Convert.ToBase64String(
122          bytevals))
123  End Sub
124
125  Sub Main(args() as String)
126      Dim input = Console.ReadLine
127      If input = "ENC" Then
128          Encode()
129          Return
130      End If
131      Dim bytes = System.Convert.FromBase64String(
132          input)

```

```

127     Dim A(2)() as Fraction
128     Dim B(2)() as Fraction
129     For y = 0 to 2
130         A(y) = new Fraction(){Nothing, Nothing,
131             Nothing}
132         For x = 0 to 2
133             Dim num_a as Int16 = BitConverter.
134                 ToInt16(bytes,x*4+y*3*4)
135             Dim den_a as Int16 = BitConverter.
136                 ToInt16(bytes,x*4+y*3*4+2)
137             Dim frac = new Fraction(num_a, den_a)
138             A(y)(x) = frac
139         Next
140     Next
141     For y = 0 to 2
142         B(y) = new Fraction(){Nothing, Nothing,
143             Nothing}
144         For x = 0 to 2
145             Dim num_b as Int16 = BitConverter.
146                 ToInt16(bytes,x*4+y*3*4+36)
147             Dim den_b as Int16 = BitConverter.
148                 ToInt16(bytes,x*4+y*3*4+38)
149             Dim frac = new Fraction(num_b, den_b)
150             B(y)(x) = frac
151         Next
152     Next
153     Dim C(1)() as Fraction
154     Dim D(1)() as Fraction
155     For y = 0 to 1
156         C(y) = new Fraction(){Nothing, Nothing,
157             Nothing}
158         For x = 0 to 1
159             Dim num_a as Int16 = BitConverter.
160                 ToInt16(bytes,x*4+y*2*4+72)
161             Dim den_a as Int16 = BitConverter.
162                 ToInt16(bytes,x*4+y*2*4+72+2)
163             Dim frac = new Fraction(num_a, den_a)
164             C(y)(x) = frac
165         Next
166     Next

```

```

158     For y = 0 to 1
159         D(y) = new Fraction() {Nothing, Nothing,
160             Nothing}
161         For x = 0 to 1
162             Dim num_b as Int16 = BitConverter.
163                 ToInt16(bytes, x*4+y*2*4+88)
164             Dim den_b as Int16 = BitConverter.
165                 ToInt16(bytes, x*4+y*2*4+88+2)
166             Dim frac = new Fraction(num_b, den_b)
167             D(y)(x) = frac
168         Next
169     Next
170     Dim MList as new List(Of List(Of Fraction))
171     For y = 0 to 2
172         Dim nList as New List(Of Fraction)
173         MList.Add(nList)
174         For x = 0 to 2
175             nList.Add(A(y)(x))
176         Next
177     Next
178     Dim AMatrix = New Matrix(MList)
179     MList = new List(Of List(Of Fraction))
180     For y = 0 to 2
181         Dim nList as New List(Of Fraction)
182         MList.Add(nList)
183         For x = 0 to 2
184             nList.Add(B(y)(x))
185         Next
186     Next
187     Dim BMatrix = New Matrix(MList)
188     Console.WriteLine("Matrix_A:")
189     Console.WriteLine(AMatrix)
190     Console.WriteLine("Matrix_B:")
191     Console.WriteLine(BMatrix)
192     Dim M1Matrix = AMatrix * BMatrix
193     Console.WriteLine("Matrix_M1:")
194     Console.WriteLine(M1Matrix)
195     MList = new List(Of List(Of Fraction))
196     For y = 0 to 1
197         Dim nList as New List(Of Fraction)

```

```

195         MList.Add(nList)
196     For x = 0 to 1
197         nList.Add(C(y)(x))
198     Next
199 Next
200 Dim CMatrix = New Matrix(MList)
201 MList = new List(Of List(Of Fraction))
202 For y = 0 to 1
203     Dim nList as New List(Of Fraction)
204     MList.Add(nList)
205     For x = 0 to 1
206         nList.Add(D(y)(x))
207     Next
208 Next
209 Dim DMatrix = New Matrix(MList)
210 Dim M2Matrix = CMatrix*DMatrix
211 Console.WriteLine("Matrix_C: ")
212 Console.WriteLine(CMatrix)
213 Console.WriteLine("Matrix_D: ")
214 Console.WriteLine(DMatrix)
215 Console.WriteLine("Matrix_M2: ")
216 Console.WriteLine(M2Matrix)
217 End Sub
218 End Module

```

Al ejecutarlo con nuestro string, se da el siguiente resultado

```

1 >kouta@koutas-lair ~/matrix_mult (git)-[master] %
  dotnet run
2 >
  AQABAAMAAQAAAAEABQABAAGAAQAMAAEACQABAAQAAQAKAAEAAQFnAOgAZwAXAmcA7
  /9nAD8CZwDaAWcA2//OAHX9zgCH/
  M4ABQABAawAAQAIAAEAAAABAAMAAgABAAGAFwAYAJsAYAA=
3 Matrix A:
4 1      3      0/0
5 5      8      12
6 9      4      10
7
8 Matrix B:
9 257/103 232/103 535/103
10 -17/103 575/103 474/103

```


11	$-37/206$	$-651/206$	$-889/206$
12			
13	Matrix M1:		
14	2	19	19
15	9	18	11
16	20	11	22
17			
18	Matrix C:		
19	5	12	
20	8	0/0	
21			
22	Matrix D:		
23	$3/2$	$1/8$	
24	$23/24$	$155/96$	
25			
26	Matrix M2:		
27	19	20	
28	12	1	