Algorithm 1 Counting the Number of. Egistry entires containing '*Exploit*'

```
Input   InputData
Output  ResultData
1       sc = spark.SparkContext()
2       rdd = sc.parallelize(InputData )
3       rdd.flatMap(lambda x: x.split('\n'))
4       rdd.filter(lambda x : 'Exploit' in x)
5       ResultData = rdd.count()
```

Algorithm 2 *ConvertRegEntryToNestedKeyValue()*

```
Input   RegEntry
Output  RegNestedKeyValue
1       keys = RegEntry .split('=')[0].split('\')
2       value = RegEntry. split('=')[1]
3       reg = keys + value
4       whlie type(reg[0]) != dict :
5         regValue = reg.pop()
6         regKey = reg.pop()
7         nestedValue = { regKey : regValue }
8         RegNestedKeyValue.append(nestedValue)
```

Algorithm 3 *MergeRegNestedEntries()*

Input *RNE1, RNE2*

Output *RNE1+2*

```
1   keys = []
2   while True:
3     if RNE1.keys() == RNE2.keys():
4       keys.append(RNE1.keys())
5       RNE1 = RNE1.values()
6       RNE2 = RNE2.values()
7     else:
8       break
9   ListRegEntries = RNE1.values().update(RNE2.values())
10  while keys:
11    commonPath = { keys.pop() : commonPath }
12  RNE1+2 = { commonPath : ListRegEntries }
```

Algorithm 4 *ComparingRegEntries()*

```
Input   RegNestedEntry, RegRepository
Output  CompResult
regTemp = RegNestedEntry
while True:
  // forensic for Registry Key
  key = regTemp.keys()[0]
  if key:
    if key in RegRepository.keys():
      RegRepository = RegRepository[key]
      regTemp = regTemp[key]
    else:
      CompResult = RegNestedEntry
      break
  // forensic for Registry Value
  else:
    value = regTemp
    if RegRepository != value:
      CompResult = RegNestedEntry
    else:
      break
```

Algorithm 5

Input

Output

1

2

3

4

5

6

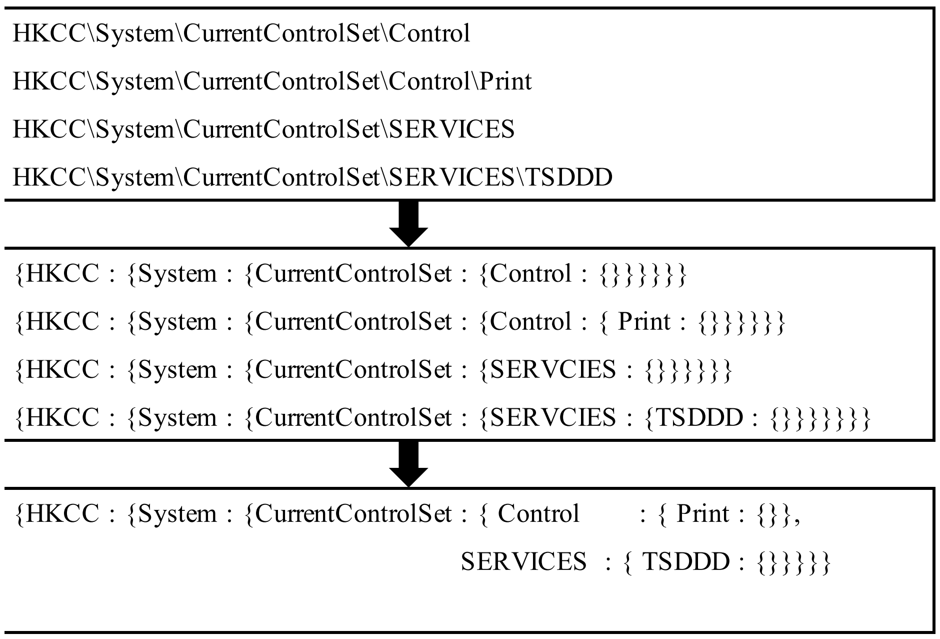
Line 4

Line 5

Line 6

Loading Windows Registry into HDFS

```
ExportedData, nPartitions
regRDD
sc = spark.SparkContext()
rdd = sc.parallelize(InputData )
rdd.repartition(nPartitions)
rdd.flatMap(lambda x: x.split('\n'))
rdd.map(lambda x : ConvertRegEntryToNestedKeyValue (x))
regRDD = rdd.reduce(lambda x,y : MergeRegNestedEntries (x,y))
```

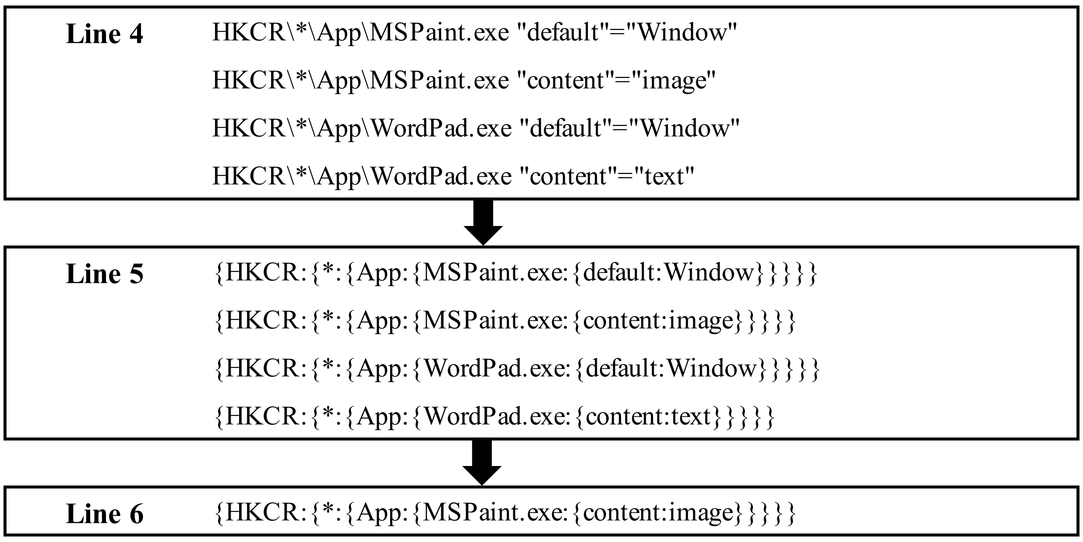


```
HKCC\System\CurrentControlSet\Control
HKCC\System\CurrentControlSet\Control\Print
HKCC\System\CurrentControlSet\SERVICES
HKCC\System\CurrentControlSet\SERVICES\TSDDD
```

```
{HKCC : {System : {CurrentControlSet : {Control : {}}}}}}
{HKCC : {System : {CurrentControlSet : {Control : { Print : {}}}}}}
{HKCC : {System : {CurrentControlSet : {SERVCIES : {}}}}}}
{HKCC : {System : {CurrentControlSet : {SERVCIES : {TSDDD : {}}}}}}}
```

```
{HKCC : {System : {CurrentControlSet : { Control       : { Print : {}},
                                         SERVICES  : { TSDDD : {}}}}}
```

```
Algorithm 6   ForensicForTargetRegKey()
Input         ExportedData, nPartitions, TargetRegKey
Output        TargetRegEntry
1             sc = spark.SparkContext()
2             rdd = sc.parallelize(InputData )
3             rdd.repartition(nPartitions)
4             rdd.flatMap(lambda x: x.split('\n'))
5             rdd.map(lambda x : ConvertRegEntryToNestedKeyValue (x))
6             TargetRegEntry = rdd.filter(lambda x : x == TargetRegKey )
```



```
Line 4   HKCR\*\App\MSPaint.exe "default"="Window"
         HKCR\*\App\MSPaint.exe "content"="image"
         HKCR\*\App\WordPad.exe "default"="Window"
         HKCR\*\App\WordPad.exe "content"="text"

Line 5   {HKCR:{*:{App:{MSPaint.exe:{default:Window}}}}}
         {HKCR:{*:{App:{MSPaint.exe:{content:image}}}}}
         {HKCR:{*:{App:{WordPad.exe:{default:Window}}}}}
         {HKCR:{*:{App:{WordPad.exe:{content:text}}}}}

Line 6   {HKCR:{*:{App:{MSPaint.exe:{content:image}}}}}
```

```
Algorithm 7
Input
Output
1
2
3
4
5
6
7
```

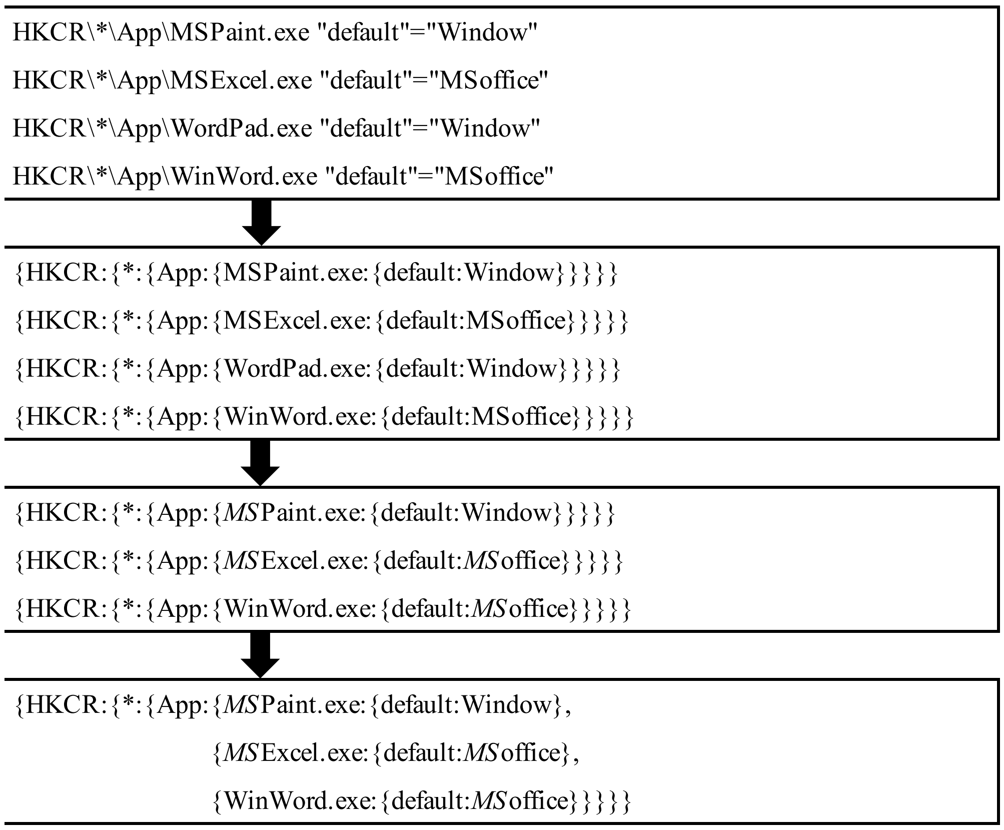
Line 4

Line 5

Line 6

Line 7

```
ForensicRegEntriesUsingKeywords()
ExportedData, nPartitions, TargetRegKeyword
RegNestedEntry
sc = spark.SparkContext()
rdd = sc.parallelize(InputData )
rdd.repartition(nPartitions)
rdd.flatMap(lambda x: x.split('\n'))
rdd.map(lambda x : ConvertRegEntryToNestedKeyValue (x))
rdd.filter(lambda x : TargetKeyword  in x)
RegNestedEntry  = rdd.reduce(lambda x,y : MergeRegNestedEntries (x,y))
```



```
HKCR\*\App\MSPaint.exe "default"="Window"
HKCR\*\App\MSExcel.exe "default"="MSoffice"
HKCR\*\App\WordPad.exe "default"="Window"
HKCR\*\App\WinWord.exe "default"="MSoffice"
```

```
{HKCR:{*:{App:{MSPaint.exe:{default:Window}}}}}
{HKCR:{*:{App:{MSExcel.exe:{default:MSoffice}}}}}
{HKCR:{*:{App:{WordPad.exe:{default:Window}}}}}
{HKCR:{*:{App:{WinWord.exe:{default:MSoffice}}}}}
```

```
{HKCR:{*:{App:{MSPaint.exe:{default:Window}}}}}
{HKCR:{*:{App:{MSExcel.exe:{default:MSoffice}}}}}
{HKCR:{*:{App:{WinWord.exe:{default:MSoffice}}}}}
```

```
{HKCR:{*:{App:{MSPaint.exe:{default:Window},
                {MSExcel.exe:{default:MSoffice},
                {WinWord.exe:{default:MSoffice}}}}}
```

```
Algorithm 8
Input
Output
1
2
3
4
5
6
7
8
9
10
11
```

```
CompareRegRepositories()
RegRepo1, RegRepo2, nPartitions
DifferentRegEntries
sc = spark.SparkContext()
rdd1 = sc.parallelize(RegRepo1 )
rdd1.repartition(nPartitions )
rdd1.flatMap(lambda x: x.split('\n'))
rdd1.map(lambda x : ConvertRegEntryToNestedKeyValue (x))
NestedRegRepo1  = rdd1.reduce(lambda x,y : MergeRegNestedEntries (x,y))
rdd2 = sc.parallelize(RegRepo2 )
rdd2.repartition(nPartitions )
rdd2.flatMap(lambda x: x.split('\n'))
rdd2.map(lambda RegEntry  : ComparingRegRepositories (NestedRegRepo1 , ConvertRegEnt
DifferentRegEntries  = rdd2.reduce(lambda x,y : MergeRegNestedEntries (x,y))
```

$yToNestedKeyValue(RegEntry))$