

Configuration Manual

MSc Research Project
Masters In Cybersecurity

Priyanka Srirangam
Student ID: X21125708

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland
MSc Project Submission Sheet



School of Computing

Priyanka Srirangam

Student Name:
X21125708

Student ID:
Masters In Cybersecurity 2022

Programme: **Year:**
Research Internship

Module:
Ross Spelman

Lecturer:
13/12/2023

Submission Due Date:
Application of Homomorphic Encryption In Open Banking

Project Title:
661 11

Word Count: **Page Count:**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:
Priyanka Srirangam

Date:
13/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Priyanka Srirangam
Student ID : X2115708

1 Introduction

This document captures the detailed information regarding necessary software and hardware components to build and execute python programs for homomorphic encryption and decryption of Sample open banking API payloads. It would help reader understand and set up similar environment as the researcher to successfully, in a practical manner.

Rest of the document has following key sections –

- Environmental setup
- Coding Implementation
- Sample Payload Encryption
- Sample Payload Decryption

2 Environmental Setup

Your second section. Change the header and label to something appropriate.

2.1 Hardware Requirements

- Available RAM – 32GB
- Minimum RAM – 8GB
- Disk space – 1 TB SSD

2.2 Software Requirements

- Windows 10
- Python 3.8.10
- C++17

2.3 Additional Tools

- Visual Studio Code

2.4 Libraries

The list of libraries used and required have been listed in the table below.

Table 1 : Libraries

Library	Description
---------	-------------

Pyfhel ¹	It is an optimized Python API library for C++ based backend for homomorphic encryption schemes such as BGV, BFV and CKKS.
Numpy ²	It is a Python library which supports large, multi-dimensional arrays and a number of mathematical operations on the arrays.
Base64 ³	This Python module offers functions for encoding binary data into printable ASCII based strings and for decoding ASCII character based strings into Binary data.
Json ⁴	It is a library used to parse JSON from string or files on the file system.

3 Coding Implementation

3.1 Encryption Programs

In the following section, screenshots are provided for Python programs built to homomorphically encrypt transaction and party API payloads based on BFV scheme, using Pyfhel and other libraries mentioned above.

3.1.1 Encryption of Transaction API using BFV

```

1  import json
2  import numpy as np
3  from Pyfhel import Pyfhel, PyPtxt, PyCtxt
4  import base64
5  from datetime import datetime
6
7
8  class my_dictionary(dict):
9      # __init__ function
10     def __init__(self):
11         self = dict()
12     # Function to add key:value
13     def add(self, key, value):
14         self[key] = value
15
16     def is_timestamp(attributeVal):
17         try:
18             datetime.strptime(attributeVal, '%Y-%m-%dT%H:%M:%S%z')
19             return True
20         except ValueError:
21             return False
22
23     def base64ofitem(data):
24         return base64.b64encode(data).decode('utf-8')
25
26     def encryptPiiIntValue(piiValArr, HE):
27         cipherPiiVal = HE.encryptInt(piiValArr)
28         decryptedPiiV1 = HE.decryptInt(cipherPiiVal)[0]
29         print("all good ", decryptedPiiV1)
30         return base64ofitem(cipherPiiVal.to_bytes())
31

```

¹ <https://github.com/ibarrond/Pyfhel>

² <https://numpy.org/doc/stable/release/1.26.0-notes.html>

³ <https://docs.python.org/3/library/base64.html>

⁴ <https://docs.python.org/3/library/json.html>

```

32 def encryptPiiIntArray(piiValArr, HE, len):
33     cipherStringVal = HE.encryptInt(piiValArr)
34     decryptedPiiVl = HE.decryptInt(cipherStringVal)[:len]
35     print("all good ", decryptedPiiVl)
36     return base64ofitem(cipherStringVal.to_bytes())
37
38 def encryptPiiStringValue(piiValueString, HE):
39
40     char_array = np.fromstring(piiValueString, dtype='S1')
41     print(list(char_array))
42     myChrToIntlist = []
43     for character in char_array :
44         chrInt = ord(character)
45         print(chrInt)
46         myChrToIntlist.append(chrInt)
47     piiChrValArr = np.array(myChrToIntlist, dtype=np.int64)
48     encryptedCharArray = encryptPiiIntArray(piiChrValArr, HE, len(myChrToIntlist))
49     return encryptedCharArray
50
51 def setHeContextAndSave(data):
52     HE = Pyfhel()
53     HE.contextGen(scheme='bfv', n=1024, t = 65537)
54     HE.keyGen()
55     s_context = HE.to_bytes_context()
56     data.update({"context": base64ofitem(s_context)})
57     s_public_key = HE.to_bytes_public_key()
58     data.update({"publicKey": base64ofitem(s_public_key)})
59     s_secret_key = HE.to_bytes_secret_key()
60     data.update({"privateKey": base64ofitem(s_secret_key)})
61     return HE

```

```

62
63 def encryptionRoutine(item, value):
64     if item in piiAttributes :
65         print("PII attribute")
66         mylist = []
67         mylist.append(value)
68         try:
69             piiValArr = np.array(mylist, dtype=np.int64)
70             return encryptPiiIntValue(piiValArr, HE)
71         except (ValueError):
72             if is_timestamp(value):
73                 ("Skipping timestamp encryption")
74             else:
75                 return encryptPiiStringValue(value, HE)
76     else:
77         return value
78
79 def recursiveEncryption(item, value):
80     if isinstance(value, dict):
81         for nestItem in value:
82             print(nestItem)
83             nestItemValue = value[nestItem]
84             recursiveEncryption(nestItem, nestItemValue)
85     else:
86         encryptedVal = encryptionRoutine(item, value)
87         for idx, itemToUpdate in enumerate(transactionData):
88             if item in itemToUpdate:
89                 transactionItem = transactionData[idx]
90                 transactionItem.update({item: encryptedVal})
91                 break
92

```

```

93
94 f = open('transaction.json')
95 data = json.load(f)
96 transactionData = data['Transaction']
97 piiAttributes = {'TransactionReference', 'TransactionInformation', 'Balance'}
98 HE = setHeContextAndSave(data)
99 for keyValue in transactionData:
100     #print(keyValue)
101     for item in keyValue:
102         value = (keyValue[item])
103         recursiveEncryption(item, value)
104
105     with open("TransactionOutput.json", "w") as outfile:
106         json.dump(data, outfile)
107 f.close()

```

3.1.2 Encryption program for Party API

```

1  import json
2  import numpy as np
3  from Pyfhel import Pyfhel, PyPtxt, PyCtxt
4  import base64
5  from datetime import datetime
6
7
8  class my_dictionary(dict):
9      # __init__ function
10     def __init__(self):
11         self = dict()
12     # Function to add key:value
13     def add(self, key, value):
14         self[key] = value
15
16 def is_timestamp(attributeVal):
17     try:
18         # Attempt to parse the input string as a timestamp
19         datetime.strptime(attributeVal, '%Y-%m-%dT%H:%M:%S%Z')
20         return True
21     except ValueError:
22         # If an exception is raised, it's not a valid timestamp
23         return False
24
25 def base64ofitem(data):
26     return base64.b64encode(data).decode('utf-8')
27
28 def encryptPiiIntValue(piiValArr, HE):
29     cipherPiiVal = HE.encryptInt(piiValArr)
30     decryptedPiiVl = HE.decryptInt(cipherPiiVal)[0]
31     #assert (decryptedPiiVl) == (piiVal), "Incorrect decryption - check"
32     print("all good ", decryptedPiiVl)
33     return base64ofitem(cipherPiiVal.to_bytes())

```

```

35 def encryptPiiIntArray(piiValArr, HE, len):
36     cipherStringVal = HE.encryptInt(piiValArr)
37     decryptedPiiVl = HE.decryptInt(cipherStringVal)[:len]
38     #assert (decryptedPiiVl) == (piiVal), "Incorrect decryption - check"
39     print("all good ", decryptedPiiVl)
40     return base64ofitem(cipherStringVal.to_bytes())
41
42 def encryptPiiStringValue(piiValueString, HE):
43     # Break down the string into character array
44     char_array = np.fromstring(piiValueString, dtype='S1')
45     print(list(char_array))
46     myChrToIntlist = []
47     for character in char_array :
48         # convert individual charatcer to corresponding ASCII integer
49         chrInt = ord(character)
50         print(chrInt)
51         # Encrypt the resulting Integer
52         myChrToIntlist.append(chrInt)
53     piiChrValArr = np.array(myChrToIntlist, dtype=np.int64)
54     encryptedCharArray = encryptPiiIntArray(piiChrValArr, HE, len(myChrToIntlist))
55     return encryptedCharArray
56

```

```

58 def setHeContextAndSave(data):
59     HE = Pyfhel()
60     HE.contextGen(scheme='bfv', n=1024, t = 65537)
61     HE.keyGen()
62     s_context = HE.to_bytes_context()
63     data.update({"context": base64ofitem(s_context)})
64     s_public_key = HE.to_bytes_public_key()
65     data.update({"publicKey": base64ofitem(s_public_key)})
66     s_secret_key = HE.to_bytes_secret_key()
67     data.update({"privateKey": base64ofitem(s_secret_key)})
68     return HE
69
70 def encryptionRoutine(item, value):
71     if item in piiAttributes :
72         print("PII attribute")
73         mylist = []
74         mylist.append(value)
75         try:
76             piiValArr = np.array(mylist, dtype=np.int64)
77             return encryptPiiIntValue(piiValArr, HE)
78         except (ValueError):
79             if is_timestamp(value):
80                 # skip timestamp encryption
81                 ("Skipping timestamp encryption")
82             else:
83                 return encryptPiiStringValue(value, HE)
84     else:
85         return value

```

```

87 def recursiveEncryption(item, value):
88     if isinstance(value, dict):
89         for nestItem in value:
90             print(nestItem)
91             nestItemValue = value[nestItem]
92             recursiveEncryption(nestItem, nestItemValue)
93     else:
94         encryptedVal = encryptionRoutine(item, value)
95         for idx, itemToUpdate in enumerate(transactionData):
96             if item in itemToUpdate:
97                 transactionItem = transactionData[idx]
98                 transactionItem.update({item: encryptedVal})
99                 break
100
101 encryptedOutput = my_dictionary()
102 f = open('party.json')
103 data = json.load(f)
104 transactionData = data['Party']
105 piiAttributes = {'Name', 'FullLegalName', 'AccountRole'}
106 HE = setHEContextAndSave(data)
107 for keyValue in transactionData:
108     for item in keyValue:
109         value = (keyValue[item])
110         recursiveEncryption(item, value)
111
112 with open("PartyOutput.json", "w") as outfile:
113     json.dump(data, outfile)
114 f.close()

```

3.2 Decryption Programs

In the following section, screenshots are provided for Python programs built to homomorphically decrypt transaction and party API payloads based on BFV scheme, using Pyfhel and other libraries mentioned above.

3.2.1 Decryption program for Party API payload

```

1 import json
2 import numpy as np
3 from Pyfhel import Pyfhel, PyPtxt, PyCtxt
4 import base64
5
6 class my_dictionary(dict):
7     # __init__ function
8     def __init__(self):
9         self = dict()
10     # Function to add key:value
11     def add(self, key, value):
12         self[key] = value
13
14 def setHEContext(HE, item, value):
15     if (item == 'context') :
16         contextBinary = base64.b64decode(value)
17         HE.from_bytes_context(contextBinary)
18     if (item == 'publicKey') :
19         publicKeyBinary = base64.b64decode(value)
20         HE.from_bytes_public_key(publicKeyBinary)
21     if (item == 'privateKey') :
22         privateKeyBinary = base64.b64decode(value)
23         HE.from_bytes_secret_key(privateKeyBinary)
24
25 def decryptionRoutine(item, value):
26     if item in piiAttributes :
27         ciphertextBinary = base64.b64decode(value)
28         ciphertextBinary = PyCtxt(pyfhel=HE, bytestring=ciphertextBinary)
29         plaintext = HE.decryptInt(ciphertextBinary)
30         print("plain text array", plaintext)
31         plaintextArr = plaintext[plaintext != 0]
32         if(len(plaintextArr) > 1) :

```



```

33         actualtext = ''.join(chr(code) for code in plaintextArr)
34         print("actual text", actualtext)
35         return actualtext
36     else :
37         return plaintextArr[0]
38     else:
39         return value
40
41 def recursiveDecryption(item, value):
42     if isinstance(value, dict):
43         for nestItem in value:
44             #print(nestItem)
45             nestItemValue = value[nestItem]
46             recursiveDecryption(nestItem, nestItemValue)
47     else:
48         decryptedVal = decryptionRoutine(item, value)
49         for idx, itemToUpdate in enumerate(transactionData):
50             if item in itemToUpdate:
51                 transactionItem = transactionData[idx]
52                 transactionItem.update({item: decryptedVal})
53                 break
54
55 f = open('PartyOutput.json')
56 data = json.load(f)
57 transactionData = data['Party']
58 piiAttributes = {'Name', 'FullLegalName', 'AccountRole'}
59 contextAttributes = {'context', 'privateKey', 'publicKey'}
60
61 # Setup Pyfhel context
62 HE = Pyfhel()
63 for item in data:
64     if (item == 'context') :
65         contextBinary = base64.b64decode(data[item])

```

```

66         HE.from_bytes_context(contextBinary)
67     if (item == 'publicKey') :
68         publicKeyBinary = base64.b64decode(data[item])
69         HE.from_bytes_public_key(publicKeyBinary)
70     if (item == 'privateKey') :
71         privateKeyBinary = base64.b64decode(data[item])
72         HE.from_bytes_secret_key(privateKeyBinary)
73 for key in contextAttributes :
74     data.pop(key)
75
76 for keyValue in transactionData:
77     for item in keyValue:
78         value = (keyValue[item])
79         recursiveDecryption(item, value)
80
81 with open("DecryptedTransaction.json", "w") as outfile:
82     json.dump(data, outfile)

```

3.2.2 Decryption program for Transaction API

```
1  import json
2  import numpy as np
3  from Pyfhel import Pyfhel, PyPtxt, PyCtxt
4  import tempfile
5  import base64
6
7  tmp_dir = tempfile.TemporaryDirectory()
8  tmp_dir_name = tmp_dir.name
9
10
11  class my_dictionary(dict):
12      # __init__ function
13      def __init__(self):
14          self = dict()
15      # Function to add key:value
16      def add(self, key, value):
17          self[key] = value
18
19  def setHEContext(HE, item, value):
20      if (item == 'context') :
21          contextBinary = base64.b64decode(value)
22          HE.from_bytes_context(contextBinary)
23      if (item == 'publicKey') :
24          publicKeyBinary = base64.b64decode(value)
25          HE.from_bytes_public_key(publicKeyBinary)
26      if (item == 'privateKey') :
27          privateKeyBinary = base64.b64decode(value)
28          HE.from_bytes_secret_key(privateKeyBinary)
29
30
31  def decryptionRoutine(item, value):
32      if item in piiAttributes :
33          ciphertextBinary = base64.b64decode(value)
34
35          ciphertextBinary = PyCtxt(pyfhel=HE, bytestring=ciphertextBinary)
36          plaintext = HE.decryptInt(ciphertextBinary)
37          print("plain text array", plaintext)
38          plaintextArr = plaintext[plaintext != 0]
39          if(len(plaintextArr) > 1) :
40              actualtext = ''.join(chr(code) for code in plaintextArr)
41              print("actual text", actualtext)
42              return actualtext
43          else :
44              return int(plaintextArr[0])
45      else:
46          return value
47
48  def recursiveDecryption(item, value):
49      if isinstance(value, dict):
50          for nestItem in value:
51              #print(nestItem)
52              nestItemValue = value[nestItem]
53              recursiveDecryption(nestItem, nestItemValue)
54      else:
55          decryptedVal = decryptionRoutine(item, value)
56          for idx, itemToUpdate in enumerate(transactionData):
57              if item in itemToUpdate:
58                  transactionItem = transactionData[idx]
59                  transactionItem.update({item: decryptedVal})
60                  break
61
62  f = open('TransactionOutput.json')
63  data = json.load(f)
64  transactionData = data['Transaction']
65  piiAttributes = {'TransactionReference', 'TransactionInformation', 'Balance', 'AccountId'}
66  contextAttributes = {'context', 'privateKey', 'publicKey'}
```

```

67 # Setup Pyfhel context
68 HE = Pyfhel()
69 for item in data:
70     if (item == 'context') :
71         contextBinary = base64.b64decode(data[item])
72         HE.from_bytes_context(contextBinary)
73     if (item == 'publicKey') :
74         publicKeyBinary = base64.b64decode(data[item])
75         HE.from_bytes_public_key(publicKeyBinary)
76     if (item == 'privateKey') :
77         privateKeyBinary = base64.b64decode(data[item])
78         HE.from_bytes_secret_key(privateKeyBinary)
79 for key in contextAttributes :
80     data.pop(key)
81
82 for keyValue in transactionData:
83     for item in keyValue:
84         value = (keyValue[item])
85         recursiveDecryption(item, value)
86
87 with open("DecryptedTransaction.json", "w") as outfile:
88     json.dump(data, outfile)

```

4 Sample Payload Encryption and Decryption

4.1 Party API Sample payloads (before encryption, after encryption, and after decryption)

4.1.1 Party API Payload Before Encryption

```

{
  "Party": [
    {
      "PartyId": "PABC123",
      "PartyType": "Sole",
      "Name": "Semiotec",
      "FullLegalName": "Semiotec Limited",
      "LegalStructure": "UK.OBIE.PrivateLimitedCompany",
      "BeneficialOwnership": true,
      "AccountRole": "UK.OBIE.Principal",
      "EmailAddress": "contact@semitotec.co.jp",
      "Relationships": {
        "Account": {
          "Id": "22289"
        }
      },
      "Address": [
        {
          "AddressType": "Business",
          "StreetName": "Street",
          "BuildingNumber": "15",
          "PostCode": "NW1 1AB",
          "TownName": "London",
          "Country": "GB"
        }
      ]
    }
  ]
}

```

4.1.2 Party API Payload After Encryption

```
1 {"Party": [{"PartyId": "PABC123", "PartyType": "Sole", "Name": "XqEQBAAAAABxQAAAAAAAAAD8fuxTTFIirCSmrQwg  
+OE0GICU0BF1Im6TFac1IEar1IAATIAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAP/  
AQAAAAAAAABeoRAEAAAAABHAAAAAAAAAAGAAAAAAAJCoUDAAAAAMPiegMAAAAIQimIAQAAAAAD9WCECAAAAAANuscAMAAAAA  
+mqSBwAAAAACISCHAAAAAPXR8gIAAAAAQXctBAAAAAOPgQAAAAAIdyIqIAAAAA8DiDAQAAAAADiedDAAAAAGfsHQMAAAAI1FdsAQAAAAAD+tn4DAAAAAEabxAAAAAA+Hp  
+BQAAAAABFIssEAAAAAGc+rQIAAAAA+XV5AAAAAACqs4QFAAAAAAP31oIAAAAAAe6t1AgAAAAATweMEAAAAACVY+QQAAAAAPsANBQAAAAAD/  
zeIDAAAAALBZKIAAAAAAPFomBwAAAAADZPtcdAAAAABJFnYAAAAAA3NBQBAAAAAAHjGAAAAAAAN7AIQEAAAAAYIPABwAAAAACaSkkGAAAAAGGgwQAAAAAFYmGAgAAAAABQNAIBAAAAAGvMlg  
UAAAAAyoySagAAAAABusbkFAAAAAAL5rwgQAAAAAC1BYAgAAAAAC0ScUAAAAAACV59gAAAAAA7ZlgBwAAAAADi4asCAAAAAAD81ZwUAAAAARPUvAAAAAABnGG8BAAAAAMkUIQAAAAAA  
+34TAQAAAAADr400HAAAAALcfwQQAAAAAEI05BgAAAAABd9y8HAAAAAN6tEAEAAAAA9mNHQAQAAAAABACMAHAAAAAMVeFwEAAAAABDv4AAAAAABRCB0DAAAAANnd4wMAAAAAAPsubAwAAAAAC7/  
7YCAAAAAEX05QUAAAAAGSABAAAAACm9E8GAAAAAP3bAAIAAAAAAkjnUAAAAAACea  
+YCAAAAAIoJagYAAAAAKrPRAQAAAAAB6ItkAAAAABvDeAEAAAAAI25HagAAAAADsQJ8HAAAAAIKwVAAAAAAOwlu  
+AAAAAAAJIoICAAAAAFZgeQAAAAAQMT6AQAAAAAB2uh8GAAAAAHtdDdQIAAAAAATmdBgAAAAADYLOQDAAAAAAHVHgIAAAAAANhWJAAAAAADxwFkEAAAAABU7rgCAAAAA/  
3rBBGAAAAAfrpCAAAAAACVXIATIAAAAAIjnnBAAAAAC73WIDAAAAAC8sIQYAAAAAEsEKAQAAAAACenb0BAAAAADEJcwUAAAAAJgViAwAAAAAx4tkBAAAAAHvOKgMAAAAAAbfbgBQAAAAAJem4  
CAAAAAALym9gQAAAAAArkGbbAAAAABBPxoGAAAAAANlDDgIAAAAAAmnoABgAAAAADwcyIDAAAAAAIImxYAAAAAAqBXWAgAAAAAFJesFAAAAAANowUJAUAAAAAonJlBQAAAAADiuJoCAAAAAAN7RIQUAA  
AAA48VtBAAAAAADf4wCAAAAAACm6AUAAAAA+YBNBQAAAAAUXqsFAAAAAADXaHAEAAAAA0rqbBgAAAAABwz88FAAAAAADUMzgAAAAAA5atvAAAAAADLlwoBAAAAABiyKgUAAAAAMYR/  
BQAAAAABPx+IAAAAAAJQovAAAAAA4nMrAQAAAAABSiIHAAAAAJ2JKQAAAAAAWnCBAAAAAAAYqbCEAAAAAK  
+xoQMAAAAAAMV9QAQAAAAACVn08CAAAAAAPSCegCAAAAAArrquBAAAAAD3KsGAAAAAA1gfQAAAAAvuBgBQAAAAADTLNgFAAAAAAkaNAEAAAAAAxvJwAAAAAAVoqYHAAAAAA/  
2YQIAAAAAAGxBcgAAAAAAHzbHAAAAABP2+wQAAAAAJogbbgAAAAACsNZYFAAAAAAK69hacAAAAALxokBAAAAAB+fckAAAAAPwJTQAAAAAGNrIbgAAAAADtfcMBAAAAAHfdgYAAAAAXd/  
NAQAAAAACLSHQAAAAAAN5TxgEAAAAAIDNAAAAAABNRncHAAAAAIE4cAEAAAAA7oCSAwAAAAADXpeoEAAAAALUFxgEAAAAAKhnHBwAAAAABUINUAAAAAMfUJgEAAAAARz5eAgAAAAADwakoBA  
AAAAAL9LigQAAAAAp316BgAAAAABzRakFAAAAAATbzgEAAAAAXMRqBgAAAAADrx2kCAAAAAAC2frgcAAAAAMKzmAAAAADTQrgCAAAAAACD7QwQAAAAAf1lpBQAAAAADKNCCHAAAAAMh2OAEAAAA  
AbjJ7AwAAAAACHow8HAAAAAETTBQIAAAAAAoqs8BgAAAAAJpKUDAAAAAB/1jgUAAAAAo  
+IJBgAAAAAD1gEYFAAAAAAINIzwQAAAAAKk7ybQAAAAADAH6kFAAAAAAMX7sAIAAAAAAKwJBgAAAAABFksEHAAAAAACV0AMAAAAAASayQAwAAAAACPY8ICAAAAABMONGCAAAAAAbtb  
+AQAAAAAC2RgGAAAAADUvugIAAAAAeHntBgAAAAAC3dF8FAAAAAAJZJQAAAAAAN/MwAAAAAAAgTL4BAAAAALjrSgcAAAAAJCxpAAAAAA53c8FAAAAAAG68nQAAAAAAS14  
+BAAAAABrPwsAAAAAAI/1wQUAAAAAPiUMBgAAAAADBM8HAAAAAIEimgEAAAAAGFQUBWAAAAA3qqYFAAAAAO+V6AQAAAAAazrTlWAAAAADL634BAAAAAC/  
sRgcAAAAAAScbAQAABBJgDUCAAAAAA4xjwUAAAAAJKXqAAAAAAC3Dd4FAAAAAABX0QUAAAAANw/4AwAAAAAN8L0AAAAAGmRmQYAAAAAVpjbAAAAAA9uhUGAAAAAD  
+FBQUAAAAAZmcPCBgAAAAARkUSGAAAAAJQIRACAAAAA9hL/AQAAAAACyq/gGAAAAA0J3ZwQAAAAAovhJBQAAAAACc0IMEAAAAANprnyYAAAAADsFAwAAAAABt/  
pYBAAAAALv1WwEAAAAAJVBVBQAAAAAW/  
lEDAAAAAHx4cgQAAAAACmKXAgAAAAACiWTEGAAAAAEfdmYAAAAAwDwYAgAAAABo1gAGAAAAADA7lAQAAAAABYl4AQAAAAABHoFwEAAAAABw87gAAAAAAvIEAQAAAAADcUyWAAAAAOZw4gA  
AAAAALHvSBgAAAAAB0mKHAHAHFiHwAAAAAAJk8B8AAAAA5BP4CAAAAAEFHwYAAAAAGefwQAAAAAArZ1EFAAAAAALuNswYAAAAAG5XuAQAAAAAe4c8BAAAAAEHv6AQAAAAAQTSAwAAA  
AA8RI8AAAAAGmiKwMAAAAAKx1eBwAAAAABBg/  
4CAAAAAAFKdxAAAAAADxVGwAAAAAAlY68HAAAAAEZNdWYAAAAAEoqHBQAAAAABtaZICAAAAAEfJAAIAAAAAAUkB0BgAAAAAB3d2gEAAAAAKVgFMAAAAAANfYFAAAAAABxv/  
gAAAAAEea1wcAAAAAJz9UwAAAAAAD6
```

4.1.3 Party API Payload After Decryption

```
1 {"Party": [{"PartyId": "PABC123", "PartyType": "Sole", "Name": "Semiotec", "FullLegalName": "Semiotec Limited", "LegalStructure": "UK.OBIE.  
PrivateLimitedCompany", "BeneficialOwnership": true, "AccountRole": "UK.OBIE.Principal", "EmailAddress": "contact@semiotech.co.jp",  
"Relationships": {"Account": {"Id": "22289"}}, "Address": [{"AddressType": "Business", "StreetName": "Street", "BuildingNumber": "15",  
"PostCode": "NW1 1AB", "TownName": "London", "Country": "GB"}]}]}
```

4.2 Transaction API Sample payloads (before encryption, after encryption, and after decryption)

4.2.1 Transaction API Payload Before Encryption

```
{ Transaction.json > [ ] Transaction > { } 0 > { } Amount  
1 {  
2   "Transaction": {  
3     {  
4       "AccountId": "22289",  
5       "TransactionId": "123",  
6       "TransactionReference": "Ref 1",  
7       "Amount": {  
8         "Amount": "10.00",  
9         "Currency": "GBP"  
10      }  
11    },  
12    "CreditDebitIndicator": "Credit",  
13    "Status": "Booked",  
14    "BookingDateTime": "2017-04-05T10:43:07+00:00",  
15    "ValueDateTime": "2017-04-05T10:45:22+00:00",  
16    "TransactionInformation": "Cash from Aubrey",  
17    "BankTransactionCode": {  
18      "Code": "ReceivedCreditTransfer",  
19      "SubCode": "DomesticCreditTransfer"  
20    },  
21    "ProprietaryBankTransactionCode": {  
22      "Code": "Transfer",  
23      "Issuer": "AlphaBank"  
24    },  
25    "Balance": {  
26      "Amount": {  
27        "Amount": "230.00",  
28        "Currency": "GBP"  
29      }  
30    },  
31    "CreditDebitIndicator": "Credit",  
32    "Type": "InterimBooked"  
33  }  
}
```

4.2.2 Transaction API Payload After Encryption

```
1 [{"Transaction": [{"AccountId": "22289", "TransactionId": "123", "TransactionReference": "XqEQBAAAAABxQAAAAAAAAAD8fuxtTFLirCsmrQmg
+OEOGICU0Fb1Im6TFac1IEar1AAIAAAAAAAAAAQAQAAAAAAAAABAAAAAAAAAAAAAAAAAPPA/
AQAAAAAAAAABeORAEAAAAAAAAABAAAAAAAAAD1H8IDAAAAAOfHAEAAAAAY3xbBgAAAAADu1AGAAAAAJ9NGAMAAAA
+b8YBwAAAAChmr4AAAAAAIczjAEAAAAAP6DsAwAAAAAD7F6GAAAAAEqL3wMAAAAAUVC7BQAAAAABotSUBAAAAACjMigYAAAAAQWavBwAAAAAD12oQBAAAAAPk1GUAAAAAA5K3+AAAAADhn
+oFAAAAAEHuZQAAAAAP74nAwAAAAADw1cYCAAAAAANIYwAAAAAA/6jQAAAAADe9q4HAAAAAGM3dgUAAAAAlbJyAAAAAC3+doHAAAAALe5TwYAAAAADfQyBQAAAAAD3SesDAAAAAA/
dJQcAAAAA9u5ZBQAAAAADzrVAFAAAAAA9nwFAAAAAACY7bBwAAAAADGkcOEAAAAATFjKAcAAAAAVQg9BQAAAAABLKSEBAAAAAFPKsGUAAAAAA22xkgAAAAABWpZsEAAAAAJWqUGYAAAAAU0mkB
wAAAAACZLQCAAAAAAD71qWAAAAAAVn5oAgAAAAABJozgAAAAAAI69FwMAAAAAAU1qKAQAAAAACQYUIGAAAAANSi8AEAAAAAVx6jAAAAABoxQUCAAAAAAF/f
+gTAAAAAJ3W6AgAAAAAC0jW4HAAAAAP6m0QMAAAAAAtbaAgAAAAAC5XC8EAAAAAJyWkAQAAAAAC9u8AAAAABf0Q0AAAAAAEJeqiwUAAAAADChABAAAAA8k8TAAAAAACCU
+uAAAAAAUJeXQAAAAAABUuHFAAAAAALE9nAEAAAAARza6AwAAAAADe6gYDAAAAAAN711wYAAAAATQn+AgAAAAABSYwCAAAAATQYkAYAAAAAY5G1BwAAAAADbX0kAAAAAAMFB/
QMAAAAAyxbBAAAAAD7F6UCAAAAAB+jHgQAAAAAHmKCBAAAAAABe9wAEAAAAAG17IQAAAAAS1B1BQAAAAAozd8CAAAAAIjTigYAAAAALcreAwAAAAADbi
+8DAAAAANIgdgQAAAAA5h1OBwAAAAACKJXIGAAAAADuo1gHAAAAAAV+GBQAAAAADyd
+8BAAAAAH59uWUAAAAAJVReAQAAAAAoFiWAAAAAMC0aQMAAAAAAJKp4AQAAAAAHp2kGAAAAADAHigEAAAAAFHxKBAAAAAD87eAGAAAAABXEAwAAAAADQLzBQAAAAADyVnACAAAAATIZQAU
AAAAAex5JAQAAAAABHhooCAAAAAADCrVQAAAAAA4d4HBAAAAABDswHAAAAAKmutQIAAAAAAYjJAwAAAAABl7EDAAAAACo5BAMAAAAA3j7IAAAAAAHyk8EAAAAAFw6wEAAAAAHIVQBwAAA
AD/jw4GAAAAADbksQEAAAAAack8BwAAAAABG6vYEAAAAAHucGgMAAAAAA2vZQBAAAAAAAgvj0GAAAAAijfAcAAAAAUtZKAgAAAAABYxgAAAAAAG4uKGEAAAAAJa5oBQAAAAAD/
ZQcAAAAAAPabvWIAAAAAANBj
+BAAAAASopQFAAAAAEGHwQCAAAAAYMjVAAAAABePtAHAAAAACrCAUAAAAA0r10AgAAAAABuKA4AAAAAAI4EpwAAAAAA0dyDagAAAAASQVEEAAAAANaVpMAAAAAAZS0jBwAAAAAG9Y4AA
AAAAANISQAAAAAVcXAAAAAAACJnMEAAAAACImmwcAAAAAMhTTBgAAAAADXB/IEAAAAAovSbMAAAAAAZnw4BwAAAAAD/
hSgAAAAAAHFbbgUAAAAAKYzJAgAAAAASoXsAAAAAIIOpwcAAAAAB9TqAgAAAAACFodQGAAAAAANI0YQQAAAAATNeOQAAAAABdv
+AAAAAANnzWMAAAAAASPRuBgAAAAABgAhkGAAAAAIIQKwQAAAAAALUnAAAAAABYlYVDAAAAAGoNHQMAAAAAA2vIMBwAAAAACX0B0FAAAAAI22cQMAAAAAAXLgJAgAAAAADodCACAAAAANJXNQA
AAAAANUcWQAAAAAD
+UGABAAAAAFxhagYAAAAAP336BQAAAAABceFEBAAAAAALiYKAcAAAAAwwM1BgAAAAATXdcAAAAAFAx7xgYAAAAAAGDBAAAAACgzQUAAAAAALDBhQCAAAAAN3mXAgAAAAAD3OmACAAAAADqI0
gAAAAAAG8jXBAAAAACSkpIAAAAAAG05ZQEAAAAATvcGBwAAAAADYqTcFAAAAAAC7gYAAAAADsFyAAAAAACv2o4GAAAAAC9RxlAAAAAND/
GBgAAAAADHJkFwAAAAAMR0wMAAAAAAMwFeBAAAAADf9ZMEAAAAA07PtWMAAAAA4Um1AQAAAAADaY1IGAAAAA0wY1RAEAAAAA6J7KBQAAAAADJ7/
0AAAAAAJXJoQAAAAA22ZWAQAAAAABaFgCHAAAAAIIUWgYAAAAAMRNQAQAAAAA0y14FAAAAAEPVQWIAAAAAA
+rFBAAAAACIATABAAAAAUtUjUwEAAAAAU14UAQAAAAAC35AFAAAAATVcNAAAAAA4obAQAAAAADhJTCAAAAAADPZcAYAAAAAFiwrBwAAAAACHiRYAAAAABJ2EYAAAAAAZ9uBAAAAABT03A
EAAAAAE//AAAAAAARb5MAwMAAACk5iKEAAAAAHbVwUAAAAAv9xTBQAAAAAB/1bTFAAAAAAC4KcAAAAAjj9HAAAAAABRSBcAAAAAKM1JAEAAAA
+9p8AQAAAAAEyF7WUAAAAAZCf2AwAAAAADeDad4BAAAAAPm0LwcAAAAAIIrfAAAAAADCD8BFAAAAAADzmvgCAAAAAAr1gHAAAAAALKL EGAAAAA0IwHQAAAAA06gstBgAAAAAGeUMFAAAAABJcW
gEAAAAAU44UBwAAAAACn0kkFAAAAAEAJwAUAAAAADGLTAQAAAAABr0PwEAAAAAN1bQwMAAAAAA1o8LBgAAAAABwLykBAAAAANZrZAYAAAAA0YHIAQAAAAABRhIgAAAAAAGKbCAEAAAAAKHmaAAA
AADRwISdAAAAAKCJoAQAAAAAARkwBQAAAAAD0K08DAAAAA0peJQIAAAAAAuhkBwAAAAABZ5MgHAAAAAAN4qcqAAAAAm4o1BgAAAAADgj0wAAAAAAKHJpwIAAAAAAG7WoBAAAAACoSwKEAAAAA
MQ7haQAAAAAJRMUBAAAAADGI
+CBAAAAALpa3WiAAAAAXJ5AwAAAAAC4NDEDAAAAAAKxohQCAAAAACaBJAwAAAAABjVNEEAAAAAFDyfaIAAAAAAYKg6AwAAAAAP4bwBAAAAAFR7jJAQAAAAA1cnHBwAAAAABM1vMAAAAAAOR24wI
AAAAALxE9BwAAAAACWD+YAAAAAAT5YgMAAAAAAGa+TBwAAAAAALrEBAAAAAE1pGAYAAAAAerXKAgAAAAcO
```

4.2.3 Transaction API Payload After decryption

```
1 [{"Transaction": [{"AccountId": "22289", "TransactionId": "123", "TransactionReference": "Ref 1", "Amount": "230.00", "CreditDebitIndicator":
"Credit", "Status": "Booked", "BookingDateTime": "2017-04-05T10:43:07+00:00", "ValueDateTime": "2017-04-05T10:45:22+00:00",
"TransactionInformation": "Cash from Aubrey", "BankTransactionCode": {"Code": "ReceivedCreditTransfer", "SubCode": "DomesticCreditTransfer"},
"ProprietaryBankTransactionCode": {"Code": "Transfer", "Issuer": "AlphaBank"}, "Balance": {"Amount": {"Amount": "230.00", "Currency": "GBP"},
"CreditDebitIndicator": "Credit", "Type": "InterimBooked"}}]]
```