**Name: Andre Freeman**

**Student Id: 610921313**

**Cryptography Final Project**

Goal**:** Format Preserving Encryption (FPE) Schema for Taiwan ID number

MainIdea**:** Luby-Rack-off 3-level construction in t space using truncated AES (**OFB mode**) block for PRF.

Additional Information:

“t” is AES\_block size (128)<t<s

s is total number of options of Taiwan Id in Taiwan ID number space.

“t” is selected as 36 since 2^36 bits is needed for our special-made transformation of Taiwan ID number to bits

Language: Python

**Taiwan ID space**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Weights ‘w’** | 1 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 1 |  |  |
|  | A | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  |  |
| Value ‘v’ | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **Sum** | **%10==0** |
| w[i]\*v[i] | 1 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 10 | True |

A number is said to be in Taiwan ID space if and only if after do the above calculation shown above the sum%10==0.

Carried out by isValid()

**Special-Made Transformation**

Given an ID number “A100000001”

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Type |  | | | | | | | | | | |
| Max\_bit per section(in binary) | 11 | 1001 | 1 | 1001 | 1001 | 1001 | 1001 | 1001 | 1001 | 1001 | 1 |
|  | A | | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 8 | 1 |
|  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 8 | 1 |
| Row Example | 01 | 0000 | 0 | 0000 | 0000 | 0000 | 0000 | 0011 | 0010 | 1000 | 1 |

The transformation from ID string to binary is a process of padding and then concatenating as seen in row example in the figure above.

010000000000000000000000011001010001: Making 36 bits, hence, ‘t’ is set as 36 bits. This unique transformation was created with the goal to reduce the space of ID not in the Taiwan Id Space from the tradition ASCII conversion.

Transformation and undo transformation is done by do\_trans() and undo\_trans() accordingly.

**Encryption** - LR\_E()

Input: Plain-text “pt” (string form of Taiwan ID number) EG. “A100000001”

Output: Cipher-text, IV

Steps for encryption:

1. Data transform (do\_trans()) the input to get its 36 bit binary value
2. Initialize AES cipher block in OFB mode and obtain iv

IV would be generated by pseudo-random number

1. Spilt the input “pt” into left and right.

Thus dividing it into 16 bit for the right and left parts

NOTE: Python we use n mod 2^t/2 and n / 2^t/2 to split and truncate s

1. Loop 3 times to simulate 3 rounds Luby-Rack-off construction
   1. For each iteration except the first swap left and right

*NOTE: Code is written in this way since, swap is not done before initially round and not done after the final round*

* 1. PRF Function
     1. Uses an AES block for encryption, where the data is padded from the right (128-t) times
     2. And the result after encryption is truncated to keep the least significant t bits
     3. truncated version is returned
  2. Obtain F from PRF function applied to the ride(side) and XOR the answer with the left

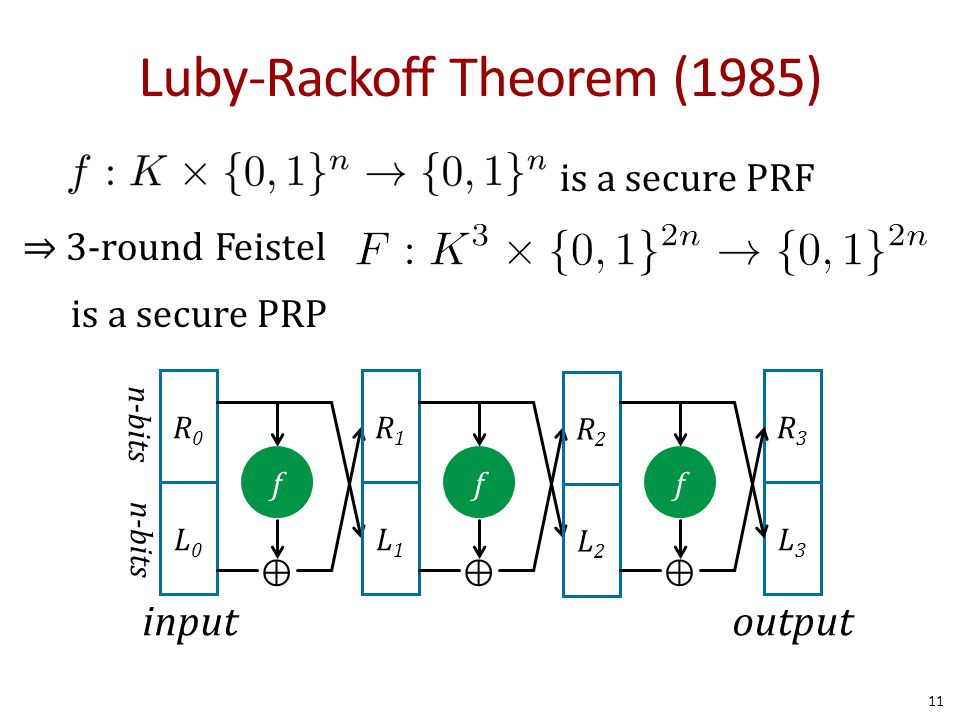


Figure 1- Showing Luby-Rackoff 3round construction where f is the obtain from the PRF function

1. Recombine the left and right parts

Take left shift << t/2 bits and XOR answer with right

1. Undo transformation (undo\_trans())
2. Check if the ID is in the Taiwan ID Space (isValid(str))

If not Valid:

* 1. Repeat steps 3 to 6

1. Return ID, iv

**Decryption** – LR\_D()

Input: Cipher-text “ct” (string form of Taiwan ID number) EG. “I106941441”

Output: Plain-text

Steps for encryption:

1. Data transform (do\_trans()) the input to get its 36 bit binary value
2. Initialize AES cipher block in OFB mode set iv for decryption mode
3. Spilt the input “ct” into left and right.

Thus dividing it into 16 bit for the right and left parts

NOTE: Python we use n mod 2^t/2 and n / 2^t/2 to split and truncate s

1. Loop 3 times to simulate 3 rounds Luby-Rack-off construction
   1. For each iteration except the first swap left and right

*NOTE: Code is written in this way since, swap is not done before initially round and not done after the final round*

* 1. PRF Function
     1. Uses an AES block for decryption, where the data is padded from the right (128-t) times
     2. And the result after encryption is truncated to keep the least significant t bits
     3. truncated version is returned
  2. Obtain F from PRF function applied to the ride(side) and XOR the answer with the left

1. Recombine the left and right parts

Take left shift << t/2 bits and XOR answer with right

1. Undo transformation (undo\_trans())
2. Check if the ID is in the Taiwan ID Space (isValid(str))

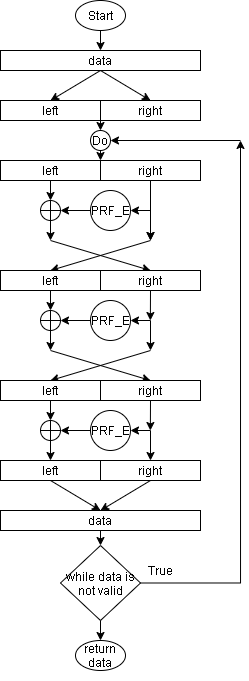
If not Valid:

* 1. Repeat steps 3 to 6

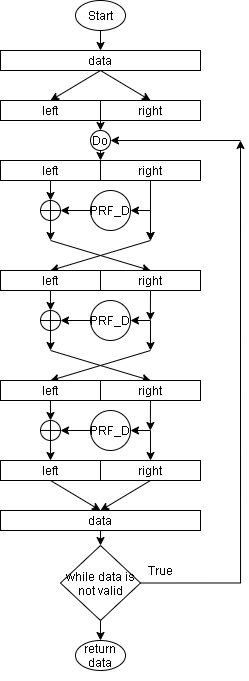
1. Return ID

**Algorithm Flowchart**

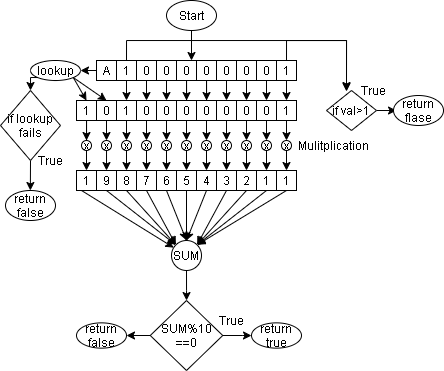
LF\_E () function



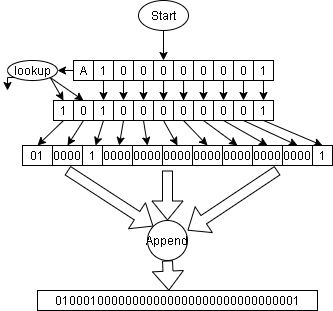
LF\_D() function



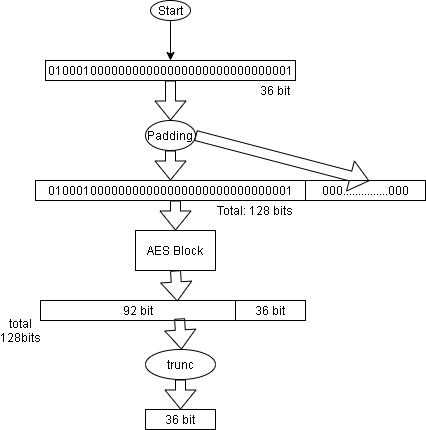
isValid() function



do\_trans() function



PRF\_E



NOTE: PRF\_D is the same however AES\_decode\_block is used.

**Results**

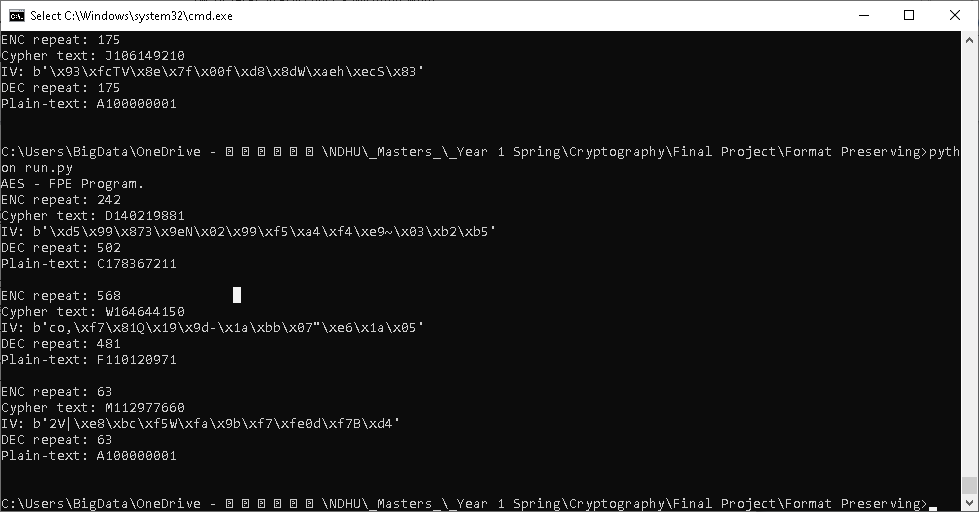


Figure - showing FPE+AES results

**Conclusion**

Please note that for my code one (1) iteration of LR\_E() and LR\_D() does not always give the correct complete encryption-decryption. The incorrect complete encryption-decryption process are considered as item that do not have 1 to 1 mapping from the Taiwan ID space to the entire AES 128bit space. These incorrect encryption-decryptions are easily identifiable when the number of times repeated for the Luby-Rackoff 3round is not the same for the encryption and the decryption as seen above in figure 2. To solve this we just redo the process with a different-new (random) IV.