|  | | |
| --- | --- | --- |
|  | |  |
|  |  | DIAL – Airtel Malawi CDR Data Anonymization |
|  |  | Process Document |
| September 5, 2018 |
|  |  | |  |  | | --- | --- | | Version No. | .1 | | Authorized by |  | |

Contact Information

| **For further information and discussions, please contact** | |
| --- | --- |
| Name |  |
| Designation |  |
| Address | Infosys Limited |
| Phone |  |
| Fax |  |
| Email |  |

Table of Contents

[Document Scope 4](#_Toc523942865)

[Requirement 4](#_Toc523942866)

[Data 4](#_Toc523942867)

[Algorithm 4](#_Toc523942868)

[Why SHA 256 5](#_Toc523942869)

[Implementation 5](#_Toc523942870)

[Pre-requisites 5](#_Toc523942871)

[Process 5](#_Toc523942872)

[Future Enhancements 6](#_Toc523942873)

[Appendix 7](#_Toc523942874)

# Document Scope

The Scope of this document is to explain the data anonymization process and algorithm that was delivered to DIAL, which enables Airtel – Malawi, to anonymize the 26 Months of RAW CDR data.

# Requirement

Airtel has a huge volume of CDR data for a period of 26 Months (2016 -2018). These data contain PII records i.e., Mobile Numbers.

This data has to be anonymized using a robust algorithm before consumed by any analytical program.

# Data

Airtel – Malawi extracts CDR data for a period of 26 Months.

Each data file contains below fields.

* USAGE\_TYPE\_NAME
* CALL\_ORIGINATING\_NUMBER
* CALL\_TERMINATING\_NUMBER
* CALL\_START\_DATE
* CALL\_START\_TIME
* LAC\_ID
* CELL\_ID

The Call Originating Number and Terminating number fields contain PII and hence the data in the two fields should be anonymized before getting consumed / moved out of the source network.

# Algorithm

We propose to use hashing technique to anonymize the data.

* A hash function is any function that can be used to map data of arbitrary size to data of a fixed size. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes.
* It takes input of a variable length sequence of bytes and converts it to a fixed length sequence. It is a one-way function.
* A key aspect of cryptographic hash functions is their collision resistance (nobody should be able to find two different input values that result in the same hash output).

To make it more secured process, we recommend SHA256 Algorithm.

# Why SHA 256

* SHA 256 is a cryptographic hash function designed by the United States National Security Agency(NSA).
* Successor to SHA-1 and is one of the strongest hash functions available
* This algorithm generates an almost-unique, fixed size 256-bit (32-byte) hash.
* Hash is a one-way function – it cannot be decrypted back. This makes it suitable for password validation, challenge hash authentication, anti-tamper, digital signatures.
* High Collision Resistance.

# Implementation

We have wrapped the SHA256 in a Python script.

# Pre-requisites

**System Configuration:**

*Storage*: Should accommodate entire raw data \* 3 times (If the Raw data is 200GB, the actual storage should be 200\*3 i.e., 600GB will be required)

RAM: 8 GB or Above

Processer: Intel® Core™ i5 -7300U CPU @ 2.60GHz 2.70 GHz equivalent or higher.

**Software Requirement:**

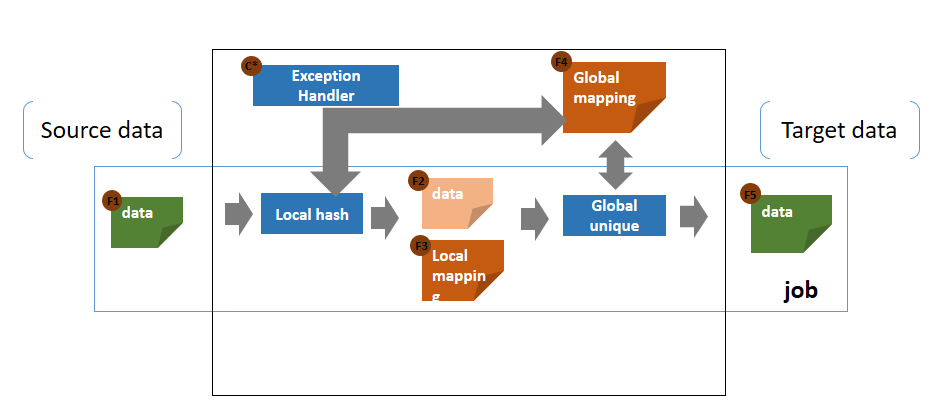
Python (version: 3.6.5 or above)

# 

# Process

The execution process explained below.

* The raw data should reside in single folder / directory.
* Python scripts should be configured to pick the source directory.
* The script executes as depicted in the below flow diagram



* In the first run, the first file will be hashed locally and will create a global mapping file in ROM.
* Global mapping file will have the hashed data across each mobile number.
* After populating the global mapping file, the job will generate Hashed Target file in different directory (e.g.: Target directory) with same name as of the source file.
* In the subsequent runs, the raw file will be compared with the global mapping and only the missed data / phone numbers will undergo hashing.
* Those data will be hashed locally (in memory) and will generate local hashed data.
* After updating the global mapping file, the new hashed target file will be generated and stored in the defined target directory.
* After successful completion of entire data anonymization, the global mapping file has to be deleted manually and the system cache should also be cleared.

# Future Enhancements

Few possible enhancements which we have thought about.

Enhancement 1:

Automating the deletion process after successful execution of data anonymization.

Enhancement 2:

Current flow is limited to sequential execution, but this can be made to run in parallel i.e., anonymization of multiple files together. But this depends on the System Infrastructure.

# Appendix

User Manual:



Unit Test document:



