**How To Ingest AES Encrypted Data With Python**

To ingest encrypted data into DWH, we may ingest the data as it is or decrypt and load it to the database, depending on the business requirements. It is always good to know how to decrypt encrypted data. There are many encryption methods. Encryption usually happens at the application (either client or server) and encrypted data get passed to the database. It is up to your business requirements to ingest the encrypted data into DWH  as it is or decrypt it before loading. There are also may ways to decrypt data in data ingestion process. SQL has its own decryption functions. You can decrypt it with ETL tools or programming language.

We will use Python here as it offers a pretty good crypto solutions which enable you to perform crypt tasks without knowing too much about how it works. To narrow out scope, we will focus on most widely used encryption algorithm, AES (Advanced Encryption Standard) encryption.

AES is secure enough (apparently US uses it for securing sensitive but unclassified material) and I have seen it is used quite often to protect customer personal information in ecommerce applications. For example, the website captures personal information on the form and the application encrypts it with AES algorithm before passing it the database. In this way, the customer's personal information is stored in the encrypted format for privacy protection. To decrypt the data, you need the key.

AES encryption is a type of [symetric-key algorithms](https://en.wikipedia.org/wiki/Symmetric-key_algorithm" \t "_blank). Encryption happens by using a key and an initialisation vector. AES is a block cipher that encrypts data on a per-block basis and uses keys that are 16, 24 or 32 bytes (128, 192, or 256 bits) and the initialization vector lengths of 16 bytes (128 bits) which can be randomly generated at encryption. You can check out the online tool to encrypt and decrypt with different methods [here](http://aes.online-domain-tools.com/). If you are interested in in-depth mechanism, [this blog article](http://www.jscape.com/blog/aes-encryption) is a good read.

AES encryption is often used for encrypting files or content in the storage device as well as for personal information for ecommerce applications. For example AWS uses 256-bit AES for the [S3 server-side encryption](https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html), default [server-side Glacier encryption](https://aws.amazon.com/glacier/faqs/) and [EBS volume encryption](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html).

The encrypted value for a personal data such as email is a string of random characters, which look like 0GSnnvEk16lFmuSVfmTRjgVQfm70pH0Cgda3Abu+LmiWmnwa88Hci2NPRl51rwHe. The first 16 letters are the initialisation vector, called IV. By using IV and the encryption key, we can decrypt it to the actual email like datahack@datahack.com.

We will use the [Crypto](https://www.dlitz.net/software/pycrypto/api/current/Crypto-module.html) package which is a great cryptography toolkit for Python. For AES, we will use [Crypto.Cipher.AES](https://www.dlitz.net/software/pycrypto/api/current/Crypto.Cipher.AES-module.html" \t "_blank). It also has a Random module to create an initialisation vector (IV) randomly for encryption.

The code examples below work for both Python 2.7 and 3.

**Scenario**

You have an encryption key and an encrypted customer email address below. The email was encrypted by 128-bit AES (AES-128). Let’s decrypt this.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

key = 'LyfZTeBjLOmjTsoq'

string = 't-QGGHNPkGR2vSXR1J7fYIfUQE7L5RBbmNkUHi9Nh8e2EV1JSe4GNbjypaX8\_o92'

[/cc]

AES128 uses first 16 characters as the initialisation vector. In the cipher function, you can use the key and the first 16 character to decrypt the rest of the string. The unpad() function will remove the white space if there are any. Urlsafe\_b64decode usually works better than just using b64decode.

Here is the function. The decrypted email address is [john.west@mydatahack.com](mailto:john.west@mydatahack.com). It can be used to decrypt any value in the column when you ingest the data.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

Now, let’s take a look at encryption. We have the email address [john.west@mydatahack.com](mailto:john.west@mydatahack.com) and encryption key, LyfZTeBjLOmjTsoq. We can use the below function to encrypt it with AES-128.

The most important part is to make the string multiple of the block size (depending on the bits of AES). The block size for AES-128 is 16. The pad() function just does that.

The initialisation vector is generated randomly. Hence, you will get a different encrypted string each time you run the function. As the first 16 letters are used for decryption, you will always get the same original value when you decrypt it.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

On a final note, encryption key can be generated randomly by using a tool like [this one](https://www.random.org/strings/). Depending on the bits, you can generate different lengths of keys. 16, 24 or 32 characters for AES-128, AES-192, or AES-256, respectively.

Awesome!