Members:- Ko Chen Chen (KXC170002)
Lauren Murphy (LAM150230)
Tanushri Singh (TTS150030)
Course:- CS 6438.001
Due:- 5/13/2019

BoxCrypt: Adding Encryption Support to Box File Sharing Service

Division of Labor Among Group Members

BoxCrypt has three modules:

1. The user interface, which consisted of a Flask server running locally on the user's machine
2. The interface between our service and Box, including
    a. Mechanisms for registering users with BoxCrypt and getting access to their account
    b. Functions for uploading, downloading and sharing files
3. Functions for encrypting and decrypting files and generating symmetric and asymmetric keys

Tanushri implemented the user interface and helped with backend, Ko Chen implemented the interface between BoxCrypt and Box, and Lauren implemented the functions for file encryption, key generation and getting access to user accounts with OAuth 2.

Major Steps of Implementation

Setting Up Environment and Choosing Tools

We chose Python 3 because it lends itself well to prototypes and offers a variety of powerful frameworks like `cryptography` and `flask`.

Requirements

There are 3 use cases:

1. (Sharer) Upload and share an encrypted file and the encrypted file key(s) for each sharee
2. (Sharee) Download encrypted file and file key and decrypt
3. Register by uploading and publicly sharing your public key.

Architecture and Design

Each of the use cases has one of the following three levels, so we split the application into the layers described in the **Division of Work** section.

1. Accepting user input
    a. E.g. sharing a file requires the user to provide a filename via the user interface
2. Preparing files for upload / processing them after download.
    a. E.g. file encryption / decryption, key generation
3. Uploading / downloading files from / to Box.

We wanted to encrypt files in such a way that Box administrators wouldn't be able to decrypt it. Encrypting large files with asymmetric keys was not feasible, but sharing a symmetric key on Box between collaborators would make it possible for administrators to decrypt the file. We therefore opted to use both symmetric and asymmetric keys. The owner of the file AKA the sharer generates a symmetric key and uses it to encrypt the file. The sharer then encrypts the symmetric file key with the public user key of the sharee and signs the encrypted key with their private key.

Implementing File Encryption, Key Generation and OAuth 2.0

I took an object-oriented approach and implemented a `UserKey` class to hold a user's `RSAPublic` public and/or `RSAPrivate` private asymmetric key and a `FileKey` class to hold the file's symmetric key, which consists of a `byte[]` nonce and a `byte[]` key. I use `cryptography`'s RSA module to provide file key integrity via signature and confidentiality via encryption. User public / private keys are 4096 bytes, which is the maximum I felt would be guaranteed to work on all platforms. The file is encrypted with AES in Galois/Counter Mode (GCM) to provide both file integrity and confidentiality. File keys are 256 bytes, which is the maximum.

```
UserKey:
      GENERATED_KEY_SIZE = 4096
      __init__(self, private, public)
      encrypt(self, plaintext)
      decrypt(self, ciphertext)
      sign(self, text)
      verify(self, text, signature)
      serialize_private(self)
      deserialize_private(serialized_private)
      serialize_public(self)
      deserialize_public(serialized_public)
      generate()
      __eq_(self, other)
      __ne__(self, other)
```

```
FileKey:
      GENERATED_KEY_SIZE = 256
      encrypt(self, unencrypted)
      decrypt(self, encrypted)
      serialize(self,
            sharee_public_key,
            sharer_private_key)
      deserialize(self,
            sharee_private_key,
            sharer_public_key)
      generate()
      __eq_(self, other)
      __ne__(self, other)
```

I first worked on generation and encryption and decryption for both FileKey and UserKey and next worked on serialization and deserialization. I finally wrote some wrapper functions read / write and encrypt / decrypt just by providing the filename and / or sharee / sharer username. These wrapper functions bridge basic read / write functions provided by Python and the functions of UserKey and FileKey. I decided to standardize file naming conventions midway through this process. While enforcing these conventions means our tool is less flexible for power users, I believe it is suitable for normal users. For example, let the filename be "re-animator.txt", the sharer username be "lam150230@utdallas.edu" and the sharee username be "muratk@utdallas.edu". The encrypted file will be "re-animator.txt**.bc**", the file key will be "re-animator.txt**.u.**muratk@utdallas.**fk**", the sharer public key will be "lam150230@utdallas.edu**.pk.pem**" and the sharee private key will be "muratk@utdallas.edu**.sk.pem**"

Box requires users to authenticate and authorize applications to access their account with OAuth 2.0. If the user navigates to the main page and we don't detect they've given us access, we automatically redirect them to Box to authenticate and authorize us. Box then redirects them to our application again with the access token attached to the request. We can then carry out our actions with the token.

Implementing Interface between BoxCrypt and Box

This interface consist of python Box sdk calls and various functions to translate what the user types in to the user interface to files being encrypting, uploading, and sharing. To allow this, for Box sdk we have 6 functions. Upload, Download, Search, Share, Update, GetLink. Each of the function are pretty self-explanatory, except Update and GetLink. Update is used for when a file with the same name is already uploaded to Box. Box does not allow file overwrites instead treats it as newer versions of the same file. Therefore we needed an Update function. GetLink is only used to download the latest public key list which is the list of all available public keys. There are two functions used to interpret user interface to functionality, uiShare and uiView. uiShare checks if the file is already on Box. If it is not, then it is the first time sharing that file and we do the encryption steps and upload. If it already on Box then we only need the file key and encrypted with the sharee's public key and share that key instead of the usual steps. uiView just lists the files that can be downloaded and it will decrypt with the appropriate keys when downloaded.

Implementing User Interface

The user interface was built using the Bootstrap platform and was interconnected with the backend using Flask Web Framework. The entire backend was programmed in Python so the integration between the back end and the front end was relatively straight forward. Front end was designed by first creating a navigation bar and each object in the navigation bar was designed in separate HTMLs to ensure that the project is independent yet has a consistent flow to it. I first began by creating a very basic frontend that had all the basic functionalities working. Then wrote the server script (server.py) that would connect the frontend website with all the backend functionalities. Once the website was performing what it is expected to, I went on to style it inorder to make the frontend more user friendly and presentable. Screenshots of what the User Interface looks like are included in the "Usage of Software Section"

Difficulties Faced During Implementation

Setting Up Environment and Choosing Tools / Developing the UI

We experienced a lot of problems trying to set up Python on one of our team member's Mac laptops. PyCharm seemed to spontaneously delete the venv it generated and Flask threw an error every time it started in debug mode. The latter was particularly problematic as she was our front-end developer. It was challenging to pick up UI design with Websites since no one on the team was too familiar with Frontend with the accumulation of websites

Implementing File Encryption, Key Generation and OAuth 2.0

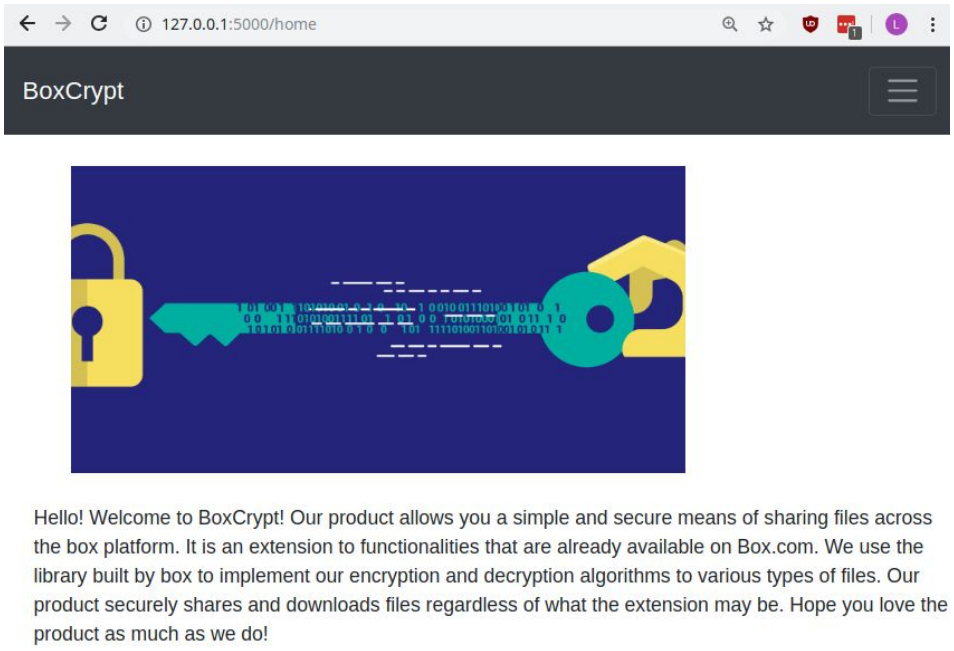
Although we intended to do a pure CLI, we figured out that since we have to collect the token from the request redirected from Box, we had to deploy a web server. We then realized it would be easier to make a web app that ran on the user's machine.

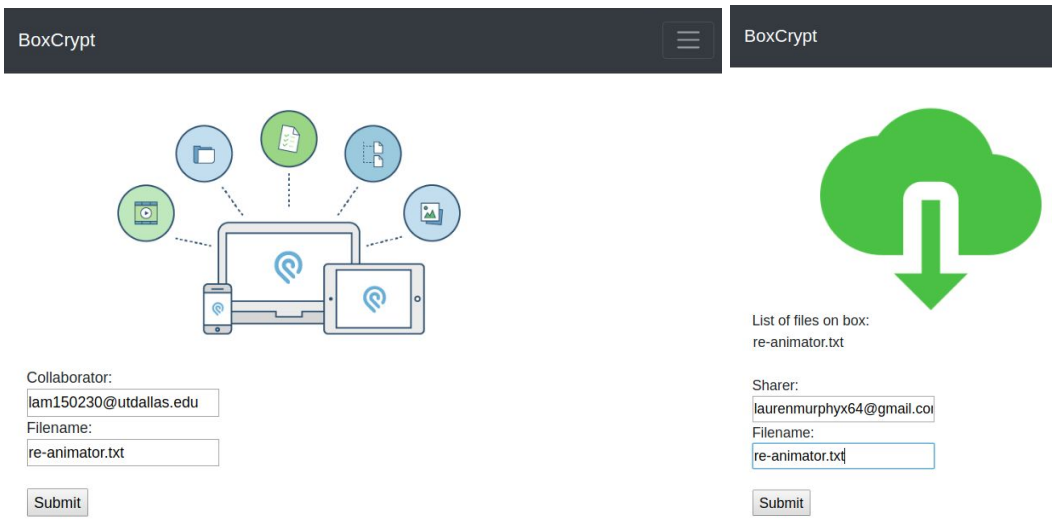
Implementing Interface between BoxCrypt and Box

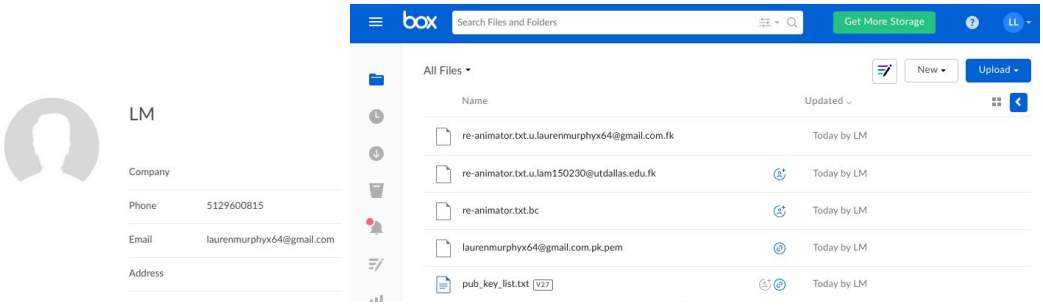
Implementing the interface requires going through the not well documented python Box sdk to find the functions that suits our need. There are simple functions for us to make use of such as upload, get_items and download. The hard part of this was the share function. Going through the documentation, there was no obvious way to share a file to a specific person on Box. So at first we decided to just get a shareable link for files and share them via an external tool like messages. However, we decided that this wasn't user friendly and digged deeper into the documentation. There we find a function that allowed us to add a collaborator to a file and was exactly what we needed.

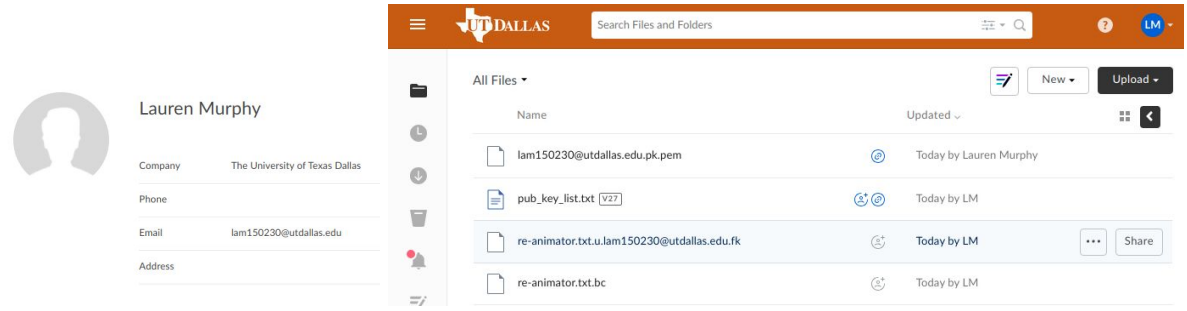
With Box and encrypt/decrypt functions done, we needed to have the user interface actually call functions to encrypt. This took us a while as we needed to make minor changes to our functions to have them work together. One problem we ran into here was, while testing, we realized that having users blindly type in the names of the files they want to download is not user friendly. (They might as well download from www.box.com at this point.) We wanted to display what files on Box are available to download directly on our user interface. This was a problem because this was our first time using flask and needed to research about how this might work.

Usage of Software with Screenshots










```
00000000  2E 2E 2E 57  65 73 74 20  68 61 64 20  ...West had
0000000C  65 6D 65 72  67 65 64 20  77 69 74 68  emerged with
00000018  20 61 20 73  6F 75 6C 20  63 61 6C 6C   a soul call
00000024  6F 75 73 65  64 20 61 6E  64 20 73 65  oused and se
00000030  61 72 65 64  2C 20 61 6E  64 20 61 20  ared, and a
0000003C  68 61 72 64  65 6E 65 64  20 65 79 65  hardened eye
00000048  20 77 68 69  63 68 20 73  6F 6D 65 74   which somet
00000054  69 6D 65 73  20 67 6C 61  6E 63 65 64  imes glanced
00000060  20 77 69 74  68 20 61 20  6B 69 6E 64   with a kind
0000006C  20 6F 66 20  68 69 64 65  6F 75 73 20   of hideous
00000078  61 6E 64 20  63 61 6C 63  75 6C 61 74  and calculat
00000084  69 6E 67 20  61 70 70 72  61 69 73 61  ing appraisa
00000090  6C 20 61 74  20 6D 65 6E  20 6F 66 20  l at men of
0000009C  65 73 70 65  63 69 61 6C  6C 79 20 73  especially s
000000A8  65 6E 73 69  74 69 76 65  20 62 72 61  ensitive bra
000000B4  69 6E 20 61  6E 64 20 65  73 70 65 63  in and espec
000000C0  69 61 6C 6C  79 20 76 69  67 6F 72 6F  ially vigoro
000000CC  75 73 20 70  68 79 73 69  71 75 65 2E  us physique.
000000D8  20 54 6F 77  61 72 64 20  74 68 65 20   Toward the
000000E4  6C 61 73 74  20 49 20 62  65 63 61 6D  last I becam
000000F0  65 20 61 63  75 74 65 6C  79 20 61 66  e acutely af
```

```
00000000  4E AE 40 95  F6 F2 9E E8  3B 6E 8E CA  N.@.....;n..
0000000C  C9 08 D9 81  C5 00 1B 8C  C4 7E 39 A5  .........~9.
00000018  EB 7A 62 31  ED 3A A7 33  07 DF 1A 35  .zb1.:.3...5
00000024  88 08 FA 35  C9 56 A0 14  0B 44 0C C7  ...5.V...D..
00000030  99 5C 82 37  D4 02 C0 00  E2 DE 1A B7  .\.7........
0000003C  22 50 3B D4  41 AC D9 7E  44 91 1E 8B  "P;.A..~D...
00000048  28 34 93 E3  E5 D1 B3 B5  8C 5A BF E8  (4.......Z..
00000054  81 A5 30 D7  40 3F C6 67  65 40 81 85  ..0.@?.ge@..
00000060  2B 69 ED 25  1E 99 BA 0E  79 C8 AB F5  +i.%....y...
0000006C  41 01 08 59  B4 BA CD C6  81 2A C5 E4  A..Y.....*..
00000078  22 0A E2 4F  EA DC 89 89  9F 34 25 B2  "..O.....4%.
00000084  8F C3 A5 D1  E8 1A 72 FC  15 ED 74 9E  ......r...t.
00000090  A4 53 2D F3  87 1B 7E 9B  BE 16 D8 C5  .S-...~.....
0000009C  B0 CF F2 E4  44 FA 93 5E  42 A5 85 8A  ....D..^B...
000000A8  28 AE FE BD  DC 2E 88 44  F0 F4 6E A7  (......D..n.
000000B4  C3 E3 29 31  25 FC BE EB  6C 41 1D 01  ..)1%...lA..
000000C0  25 2B 81 69  60 F7 75 FA  63 8D 71 B9  %+.i`.u.c.q.
000000CC  A9 8C 1F DC  4E 8B 1F 77  F3 F3 E6 87  ....N..w....
000000D8  5A B2 FB 1E  7F 5F 4B 25  85 AB 00 81  Z...._K%....
000000E4  11 DA A3 E8  E6 CE E9 3E  65 96 59 EC  .......>e.Y.
000000F0  33 99 56 1B  AE E5 B6 E3  F2 67 E7 C8  3.V......g..
```