

# Fine-grained Access Control with Attribute-based Encryption (ABE) Lab

Copyright © 2021 Anyi Liu, Xinyi Li, and Frank Wang, Oakland University and the University of Michigan.

The development of this document is funded by the following grants from the US National Science Foundation: No. 1723707 and 1623713, and Michigan Space Grant Consortium (MSGC) Research Seed Grant and Oakland University Research Seed Grant. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. If you remix, transform, or build upon the material, this copyright notice must be left intact, or reproduced in a way that is reasonable to the medium in which the work is being re-published.

## 1 Overview

This lab will help students better understand Role-based Access Control (RBAC) [1] and cryptographic technologies of Attribute-based Encryption (ABE). Students will be introduced to two different ABE: Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE) and their applications. The learning objectives of this lab are listed below:

1. Be able to describe attribute-based encryption, including CP-ABE and KP-ABE.
2. Be able to compose the security policies according to various realistic scenarios.
3. Be able to integrate security policies into crypto-keys.
4. Be able to decrypt ciphertext using correct crypto-keys.

## 2 Background

**Attribute-based encryption (ABE):** ABE is a particular category of public-key crypto-system. The private key of a specific user depends on the attributes of that user, such as age, date of birth, and position. The idea of encryption attribute was first published in Fuzzy Identity-Based Encryption and then developed as Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data.

**Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [2]:** In many situations, when a user encrypts sensitive data, she must establish a specific access control policy on *who can decrypt this data*. Suppose that the FBI public corruption offices in Knoxville and San Francisco investigate an allegation of bribery involving a San Francisco lobbyist and a Tennessee congressman. The head FBI agent may want to encrypt a sensitive memo so that only personnel with specific credentials or attributes can access it. For instance, the head agent may specify the following access structure for accessing this information: `((("Public Corruption Office" AND ("Knoxville" OR "San Francisco")) OR (management-level > 5) OR "Name:`

Charlie Eppes"). By doing this, the head agent could mean that the memo should only be read by agents who work at the public corruption offices in Knoxville or San Francisco, FBI officials very high up in the management chain, and a consultant named Charlie Eppes.

**Key-policy attribute-based encryption (KP-ABE) [3]:** KP-ABE is another class of ABE, in which ciphertext is labeled with sets of attributes and private keys are associated with access structures. The access structures control which ciphertext a user should be able to decrypt. KP-ABE has essential applications in data sharing on untrusted cloud storage.

### 3 Task 1: Lab Set-up

For this lab, we suggest you use our pre-built Docker<sup>1</sup> container, which already installed OpenABE (<https://github.com/zeutro/openabe>). OpenABE is a cryptographic library that implements a collection of attribute-based encryption (ABE) algorithms, industry-standard cryptographic functions, and tools. Use the commands listed in Figure 1 to pull and run the Docker image.

```
> docker pull yangzhou301/abe-lab
> docker run --rm -it yangzhou301/abe-lab
```

Figure 1: The command lines to pull and run the Docker image.

**Deliverable 1:** Please include a screenshot to demonstrate that the OpenABE is functioning.

## 4 CP-ABE Exercises

### 4.1 A Case Study of CP-ABE

Users List		
Name	Age	Department
Alice	24	Swimming club
Bob	21	Karate club
Cindy	25	Karate club

Before working on your lab tasks, read the following example to understand CP-ABE and OpenABE better. A confidential document about Karate is encrypted, whose content is only readable by those who belong to Karate club and older than 24. In this example, of course, only Cindy can decrypt the file, while neither Alice nor Bob can.

<sup>1</sup>Docker Desktop. URL: <https://www.docker.com/products/docker-desktop>

```
# Construct a CP-ABE crypto-system with "grizzly" as the prefix of the file name.
$ oabe_setup -s CP -p grizzly

# Generate keys for Alice, Bob, and Cindy with their attributes.
$ oabe_keygen -s CP -p grizzly -i "Age=24|Swimming-club" -o alice_key
$ oabe_keygen -s CP -p grizzly -i "Age=21|Karate-club" -o bob_key
$ oabe_keygen -s CP -p grizzly -i "Age=25|Karate-club" -o cindy_key

# Write a secret message into input.txt
$ echo "114514" > input.txt

# Encrypt the file
$ oabe_enc -s CP -p grizzly -e "((Age > 22) and (Karate-club))" -i input.txt
-o output.cpabe
```

Figure 2: The generation of private keys and encrypting a document with an access control policy

#### See Also

For more information about how to use the OpenABE command, please refer to <https://github.com/zeutro/openabe/blob/master/cli/README.md> and the following documents for details.

- [OpenABE API Guide Document](#) - explains how to install and use the library.
- [OpenABE CLI Util Document](#) - shows how to use the included command-line tools, including benchmarking.
- [OpenABE Design Document](#) - explains in detail the functionalities and algorithms implemented.

Then, you can check the result of decryption as follows:

```
# Alice decrypts with Alice's key -- should fail
$ oabe_dec -s CP -p grizzly -k alice_key.key -i output.cpabe -o alice_plain.txt

# Bob decrypts with Bob's key -- should fail
$ oabe_dec -s CP -p grizzly -k bob_key.key -i output.cpabe -o bob_plain.txt

# Cindy decrypts with Cindy's key -- should pass
$ oabe_dec -s CP -p grizzly -k cindy_key.key -i output.cpabe -o cindy_plain.txt
$ cat cindy_plain.txt
```

Figure 3: The verification of decryption the ciphertext output .cpabe

**Deliverable 2:** Please include a screenshot that demonstrates that Cindy can successfully decrypt and read `cindy_plain.txt`.

## 4.2 CP-ABE Problem Solving

As we mentioned before, a CP-ABE crypto-system specifies attributes associated with users and uses policies associated with ciphertext. A user can decrypt a ciphertext if and only if her attributes satisfy the policy. Let's use the scenarios of "*Harry Potter and the Philosopher's Stone*" to practice the CP-ABE.

Users List			
User Name	DOB	Hair Color	Level
Harry Potter	July 31, 1980	Black	Student
Ron Weasley	March 1, 1980	Blond	Student
Quirinus Quirrell	September 26, 1970	Black	Professor

Let's say a confidential document is encrypted by Hermione Granger, whose content is only viewable by Harry Potter and Ron Weasley. In other words, only Harry and Ron can decrypt and read the document, while Professor Quirrell cannot. You should create a random document for your own and demonstrate this scenario.

**Deliverable 3:** Let's say you are Hermione Granger. Please provide command lines that encrypt the document. Also, please include the screenshot(s) to demonstrate that the document has been encrypted successfully.

**Deliverable 4:** Please provide command lines that show Harry Potter and Ron Weasley can decrypt the ciphertext. Also, provide the command line(s) that shows Professor Quirrell cannot decrypt the ciphertext or decrypts with failure. You should include a screenshot(s) that demonstrates it.

## 5 KP-ABE Exercises

### 5.1 A Case Study of KP-ABE

For the KP-ABE crypto-system, the security policies are associated with users, which are specified as their private keys. The attributes are associated with ciphertext. A user can decrypt a ciphertext if and only if her private key satisfies the ciphertext's security attributes.

For example, as an employee of Amazon.com, Bob can only access emails sent to him during his employment at Amazon.com, say, from August 1 to August 31, 2019, which has been constituted in his private key. All emails circulated within Amazon.com are encrypted with some attributes, such as *from*, *to*, *date*, etc.

```
# Construct a KP-ABE crypto-system with "amazon" as the prefix of file names.
$ oabe_setup -s KP -p amazon

# Generate key slice for Bob.
$ oabe_keygen -s KP -p amazon -i "(To:Bob and (Date = Aug 1-31, 2019))"
  -o bob_KP

# Encrypt emails with different metadata in emails.
$ echo "How do you like Loki?" > input1.txt
$ oabe_enc -s KP -p amazon -e "From:Thor|To:Bob|Date=Aug 10,2019"
  -i input1.txt -o input1.kpabe

$ echo "Invitation to my small apartment this weekend." > input2.txt
$ oabe_enc -s KP -p amazon -e "From:Batman|To:Bob|Date=May 14,2021"
  -i input2.txt -o input2.kpabe

$ echo "Let's go to have a cup of coffee!" > input3.txt
$ oabe_enc -s KP -p amazon -e "From:Cindy|To:Bob|Date=Aug 14,2020"
  -i input3.txt -o input3.kpabe
```

Figure 4: The generation of private keys and encrypting a document

Then, we can check the result of decryption as follows:

```
# decrypt the first email -- should pass
$ oabe_dec -s KP -p amazon -k bob_KP.key -i input1.kpabe -o input1_plain.txt
$ cat input1_plain.txt

# decrypt the second email -- should fail (date mismatch)
$ oabe_dec -s KP -p amazon -k bob_KP.key -i input2.kpabe -o input2_plain.txt

# decrypt the second email -- should fail (receiver mismatch)
$ oabe_dec -s KP -p amazon -k bob_KP.key -i input3.kpabe -o input3_plain.txt
```

Figure 5: The generation of private keys and encrypting a document

**Deliverable 5:** Please include a screenshot that demonstrates that results of executing commands illustrated in Figure 5.

## 5.2 KP-ABE Problem Solving

Let's take an exercise from the scenes of "The Avengers" to practice KP-ABE. *The Avengers* is an organization founded by Nick Fury, the Director of S.H.I.E.L.D, on May 4, 2012. All team members are gifted superheroes committed to protecting the world from a variety of threats. Superheros

are assigned with different missions and often communicate with encrypted messages that can only be decrypted by certain receivers who are temporarily **out of** the organization HQ, which means **their secret key can only decrypt messages to themselves sent on the dates when they are not in the Avengers Tower** and prevents the secret message from being stolen by Hydra. Iron Man is the first member who joined the Avengers when it was founded. However, he disagreed with Captain America on the *Sokovia Accords* and determinedly left the Avengers on April 6, 2016. After a month, He discovered the misunderstood truth and accepted an apology from Captain America, so he returned to the group. Now, Iron Man accidentally finds four encrypted notes in Avengers Tower; their metadata is listed below:

1. The first message was sent from Thor to Hulk, dated May 10, 2012
2. The second message was sent from Black Widow to Iron Man, dated April 22, 2016
3. The third message was sent from Hawkeye to Captain America, dated May 3, 2016
4. The fourth message was sent from Captain America to Iron Man, dated on Sep 20, 2017

**Deliverable 6:** Construct a KP-ABE crypto-system and assign a secret key to Iron Man.

**Deliverable 7:** Compose the plain text of the four “proof-of-the-concept” (random) messages and then encrypt them with their metadata.

**Deliverable 8:** Which messages can be decrypted by Iron Man? Why?

## Appendices

### A Install Docker Engine on Ubuntu

The following instructions are given for those who have a difficulty of installing Docker on Ubuntu Linux. In case that you have not installed Docker successfully, please read it. The following information came from the document<sup>2</sup>. Specifically, you should enter the following commands:

---

<sup>2</sup>Install Docker Engine on Ubuntu. URL: <https://docs.docker.com/engine/install/ubuntu/>

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Figure 6: The command lines to install Docker on Ubuntu.

After that, check if Docker is installed properly by running `docker version`

```
ubuntu@ip-172-31-55-100:~$ docker version
Client: Docker Engine - Community
 Version: 20.10.7
  API version: 1.41
  Go version: go1.13.15
  Git commit: f0df350
  Built: Wed Jun  2 11:56:38 2021
  OS/Arch: linux/amd64
  Context: default
  Experimental: true

Server: Docker Engine - Community
 Version: 20.10.7
  API version: 1.41
  Go version: go1.13.15
  Git commit: f0df350
  Built: Wed Jun  2 11:56:38 2021
  OS/Arch: linux/amd64
  Context: default
  Experimental: true

docker-init:
 Version: 0.19.0
  GitCommit: de40ad0
ubuntu@ip-172-31-55-100:~$
```

Figure 7: Check if Docker is installed on Ubuntu

Add the current user to the `docker` group so that you don't have to run `docker` with `sudo`:

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
$ newgrp docker
```

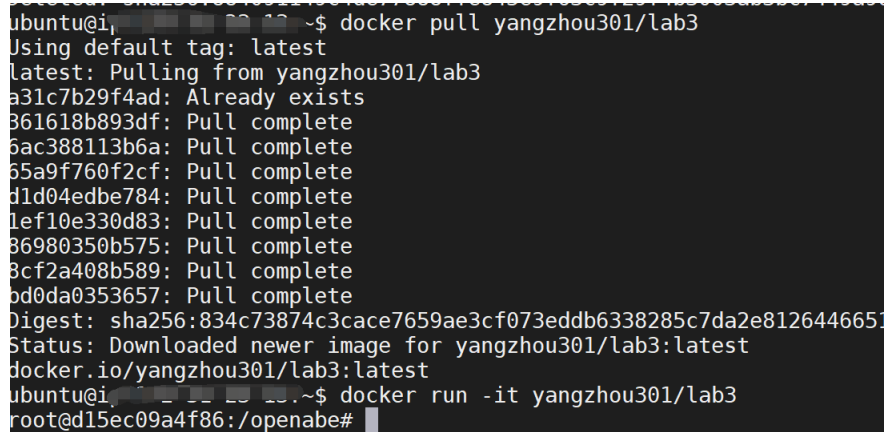
Figure 8: Add the user to the group

Then, pull the lab image and run it by:

```
$ docker pull yangzhou301/abe-lab
$ docker run -it yangzhou301/abe-lab
```

Figure 9: The command lines run the container on Ubuntu

Now, you are in the lab container:



```
ubuntu@i[REDACTED]:~$ docker pull yangzhou301/lab3
Using default tag: latest
latest: Pulling from yangzhou301/lab3
a31c7b29f4ad: Already exists
361618b893df: Pull complete
6ac388113b6a: Pull complete
65a9f760f2cf: Pull complete
d1d04edbe784: Pull complete
1ef10e330d83: Pull complete
86980350b575: Pull complete
8cf2a408b589: Pull complete
bd0da0353657: Pull complete
Digest: sha256:834c73874c3cace7659ae3cf073eddb6338285c7da2e8126446651
Status: Downloaded newer image for yangzhou301/lab3:latest
docker.io/yangzhou301/lab3:latest
ubuntu@i[REDACTED]:~$ docker run -it yangzhou301/lab3
root@d15ec09a4f86:/openabe#
```

Figure 10: Enter the lab container on Ubuntu

## References

- [1] R. Sandhu, E. Coyne, H. Feinstein, C. Youman, Role-based access control models, *Computer* 29 (2) (1996) 38–47. [doi:10.1109/2.485845](https://doi.org/10.1109/2.485845).
- [2] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (SP '07), 2007, pp. 321–334. [doi:10.1109/SP.2007.11](https://doi.org/10.1109/SP.2007.11).
- [3] V. Goyal, O. Pandey, A. Sahai, B. Waters, [Attribute-based encryption for fine-grained access control of encrypted data](https://doi.org/10.1145/1180405.1180418), CCS '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 89–98. [doi:10.1145/1180405.1180418](https://doi.org/10.1145/1180405.1180418). URL <https://doi.org/10.1145/1180405.1180418>