Data Protection Using Encryption

Lab overview

Cryptography is the conversion of communicated information into secret code that keeps the information confidential and private. Functions include authentication, data integrity, and nonrepudiation. The central function of cryptography is *encryption*, which transforms data into an unreadable form.

Encryption ensures privacy by keeping the information hidden from people who the information is not intended for. *Decryption*, the opposite of encryption, transforms encrypted data back into data; it won't make any sense until it has been properly decrypted.

In this lab, you will connect to a file server that is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance. You will configure the AWS Encryption command line interface (CLI) on the instance. You will create an encryption key by using the AWS Key Management Service (AWS KMS). The key will be used to encrypt and decrypt data. Next, you will create multiple text files that are unencrypted by default. You will then use the AWS KMS key to encrypt the files and view them while they are encrypted. You will finish the lab by decrypting the same files and viewing the contents.

Objectives

After completing this lab, you should be able to:

- Create an AWS KMS encryption key
- Install the AWS Encryption CLI
- Encrypt plaintext
- Decrypt ciphertext

Duration

This lab requires approximately **45 minutes** to complete.

Lab environment

The lab environment has one preconfigured EC2 instance named **File Server**. An AWS Identity and Access Management (IAM) role is attached, which allows you to connect to the instance by using the AWS Systems Manager Session Manager.

All backend components, such as EC2 instances, IAM roles, and some AWS services, have been built into the lab already.

Task 1: Create an AWS KMS key

In this task, you will create an AWS KMS key that you will later use to encrypt and decrypt data.

With AWS KMS, you can create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications. AWS KMS is a secure and resilient service that uses hardware security modules (HSMs) that have been validated under the Federal Information Processing Standard (FIPS) Publication 140-2, or are in the process of being validated, to protect your keys.

6. In the console, enter **KMS** in the search 🔍 bar, and then choose **Key Management Service**.
7. Choose **Create a key**.
8. For **Key type**, choose **Symmetric**, and then choose **Next**.
   ℹ *Symmetric* encryption uses the same key to encrypt and decrypt data, which makes it fast and efficient to use. *Asymmetric* encryption uses a public key to encrypt data and a private key to decrypt information.
9. On the **Add labels** page, configure the following:
   - **Alias:** `MyKMSKey`
   - **Description:** `Key used to encrypt and decrypt data files.`
10. Choose **Next**.
11. On the **Define key administrative permissions** page, in the **Key administrators** section, search for and select the check box for `voclabs` and then choose **Next**.
12. On the **Define key usage permissions** page, in the **This account** section, search for and select the check box for `voclabs` and then choose **Next**.

13. Review the settings, and then choose **Finish**.
14. Choose the link for **MyKMSKey**, which you just created, and copy the **ARN** (Amazon Resource Name) value to a text editor.

You will use this copied ARN later in the lab.

Summary of task 1

In this task, you created a symmetric AWS KMS key and gave ownership of that key to the **voclabs** IAM role that was pre-created for this lab.

Task 2: Configure the File Server instance

Before you can encrypt and decrypt data, you need to set up a few things. To use your AWS KMS key, you will configure AWS credentials on the **File Server** EC2 instance. After that, you will install the AWS Encryption CLI (aws-encryption-cli), which you can use to run encrypt and decrypt commands.

15. In the console, enter **EC2** in the search 🔍 bar, and then choose **EC2**.
16. In the **Instances** list, select the check box next for the **File Server** instance, and then choose **Connect**.
17. Choose the **Session Manager** tab, and then choose **Connect**.

18. To change to the home directory and create the AWS credentials file, run the following commands:

    ```
    cd ~
    aws configure
    ```

19. When prompted, configure the following:

    - **AWS Access Key ID**: Enter `1`, and then press Enter.
    - **AWS Secret Access Key**: Enter `1`, and then press Enter.
    - **Default region name**: Copy and paste the Region provided from the Vocareum **AWS Details** page.
    - ***Tip*** *You may need to press Ctrl+Shift+V to paste into Session Manager.*
    - **Default output format**: Press Enter.

    The AWS configuration file is created, and you will update it in a later step. The previous entries of `1` are temporary placeholders.

20. Navigate to the Vocareum console page, and choose the ℹ **AWS Details** button.

21. Next to **AWS CLI**, choose **Show**.

22. Copy and paste the code block, which starts with [default], into a text editor.

23. Return to the browser tab where you are logged in to the File Server.

24. To open the AWS credentials file, run the following command:

    ```
    vi ~/.aws/credentials
    ```

25. In the ~/.aws/credentials file, type `dd` multiple times to delete the contents of the file.

26. Paste in the code block that you copied from Vocareum.

    The AWS credentials file should now look similar to the following:

    Example of AWS credentials file contents.

    The AWS credentials file includes the following: aws_access_key_id, aws_secret_access_key, and aws_session_token. The credentials used are from the AWS Details section.

27. To save and close the file, press Escape, type `:wq` and then press Enter.
28. To view the updated contents of the file, run the following command:

    ```
    cat ~/.aws/credentials
    ```

    Now you will install the AWS Encryption CLI and export your path. By doing this, you will be able to run the commands to encrypt and decrypt data.

29. To install the AWS Encryption CLI and set your path, run the following commands:

    ```
    pip3 install aws-encryption-sdk-cli
    export PATH=$PATH:/home/ssm-user/.local/bin
    ```

Summary of task 2

In this task, you configured the AWS credentials file, which provides the ability to use the AWS KMS key that you created earlier. You then installed the AWS Encryption CLI, so that you can run encryption commands.

Task 3: Encrypt and decrypt data

In this task, you will create a text file with mock sensitive data in it. You will then use encryption to secure the file contents. Then, you will decrypt the data and view the file contents.

30. To create the text file, run the following commands:

```
touch secret1.txt secret2.txt secret3.txt
echo 'TOP SECRET 1!!!' > secret1.txt
```

31. To view the contents of the **secret1.txt** file, run the following command:

```
cat secret1.txt
```

32. To create a directory to output the encrypted file, run the following command:

```
mkdir output
```

33. Copy and paste the following command to a text editor:

```
keyArn=(KMS ARN)
```

34. In the text editor, replace **(KMS ARN)** with the AWS KMS ARN that you copied in task 1.

35. Run the updated command in the File Server terminal.

    This command saves the ARN of an AWS KMS key in the **$keyArn** variable. When you encrypt by using an AWS KMS key, you can identify it by using a key ID, key ARN, alias name, or alias ARN.

36. To encrypt the **secret1.txt** file, run the following command:

```
aws-encryption-cli --encrypt \
                     --input secret1.txt \
                     --wrapping-keys key=$keyArn \
                     --metadata-output ~/metadata \
                     --encryption-context purpose=test \
                     --commitment-policy require-encrypt-require-decrypt \
                     --output ~/output/.
```

    The following information describes what this command does:

    - The first line encrypts the file contents. The command uses the **--encrypt** parameter to specify the operation and the **--input** parameter to indicate the file to encrypt.
    - The **--wrapping-keys** parameter, and its required *key* attribute, tell the command to use the AWS KMS key that is represented by the key ARN.
    - The **--metadata-output** parameter is used to specify a text file for the metadata about the encryption operation.
    - As a best practice, the command uses the **--encryption-context** parameter to specify an encryption context.
    - The **–commitment-policy** parameter is used to specify that the key commitment security feature should be used to encrypt and decrypt.

  - The value of the **--output** parameter, *~/output/.*, tells the command to write the output file to the output directory.

  When the encrypt command succeeds, it does not return any output.

37. To determine whether the command succeeded, run the following command:

    ```
    echo $?
    ```

    If the command succeeded, the value of **$?** is **0**. If the command failed, the value is nonzero.

38. To view the newly encrypted file location, run the following command:

    ```
    ls output
    ```

    The output should look like the following:

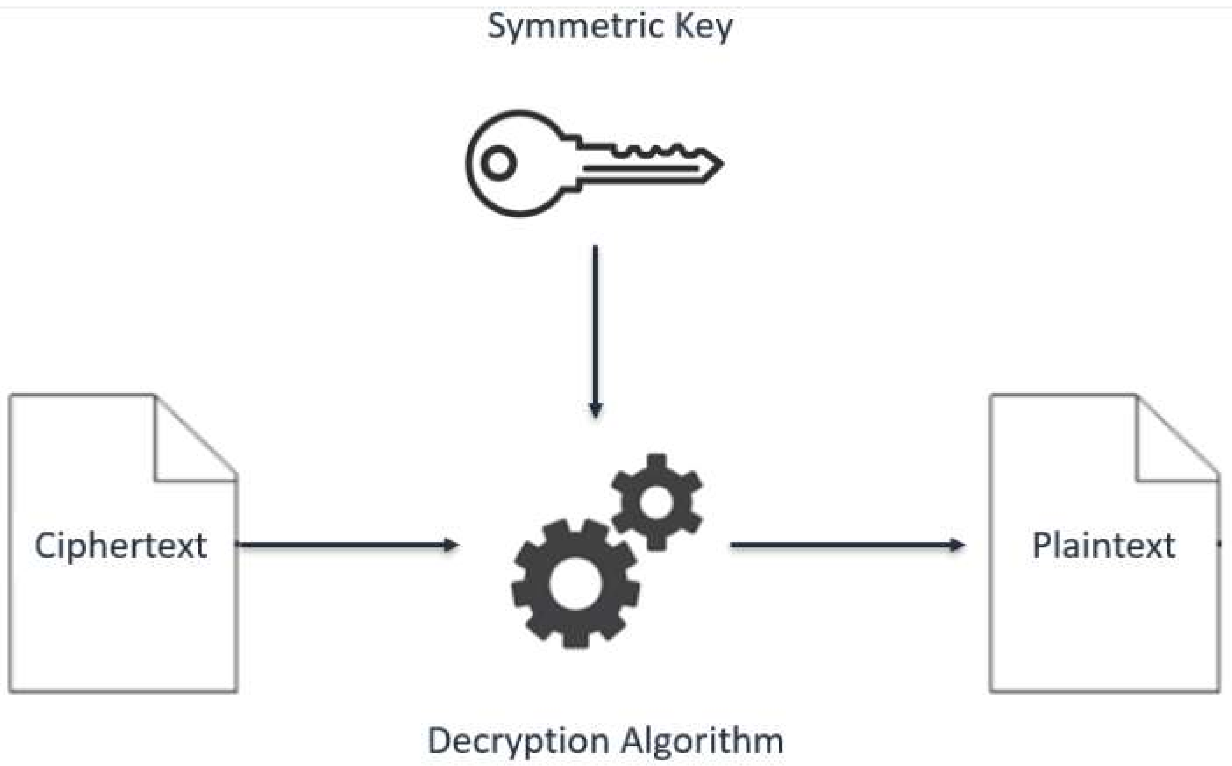
    ```
    secret1.txt.encrypted
    ```

39. To view the contents of the newly encrypted file, run the following command:

    ```
    cd output
    cat secret1.txt.encrypted
    ```

    The encryption and decryption process takes data in *plaintext*, which is readable and understandable, and manipulates its form to create *ciphertext*, which is what you are now seeing.

    When data has been transformed into ciphertext, the plaintext becomes inaccessible until it's decrypted.



This diagram shows how the same secret key and symmetric algorithm from the encryption process are used to decrypt the ciphertext back into plaintext.

Summary of task 3

In this task, you learned how to encrypt plaintext data into ciphertext by running the **--encrypt** command. You then successfully decrypted the ciphertext back into the original, readable plaintext data.